Learning with Imperfect Supervision for Language Understanding

ILLC Dissertation Series DS-2020-01

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Science Park 107
1098 XG Amsterdam
phone: +31-20-525 6051
e-mail: `illc@uva.nl`
homepage: `http://www.illc.uva.nl/`

This work is part of the ExPoSe project (NWO CI # 314.99.108), with additional support through the DiLiPaD project (NWO Digging into Data # 600.006.014) which are financed by the Dutch Research Council (NWO).

SIKS Dissertation Series 2020-03.
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Parts of the research are based on internships supported by Google LLC and Apple Inc. Views expressed in this thesis are not necessarily shared or endorsed by those funding the research.

Cover design by Reza Sedighian
Printed and bound by Ipskamp Printing
Published by IR Publications, Amsterdam

ISBN: 978-90-821695-1-5

Learning with Imperfect Supervision for Language Understanding

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in de Agnietenkapel
op vrijdag 28 februari 2020, te 10.00 uur

door

Mostafa Dehghani

geboren te Esfahan

Promotiecommisie

Promotores:	Dr. ir. J. Kamps	Universiteit van Amsterdam
	Prof. dr. M. de Rijke	Universiteit van Amsterdam
Overige leden:	Prof. dr. W.B. Croft	University of Massachusetts Amherst
	Dr. S. Gouws	DeepMind
	Dr. C. Hauff	TU Delft
	Prof. dr. A. Betti	Universiteit van Amsterdam
	Prof. dr. E. Kanoulas	Universiteit van Amsterdam
	Prof. dr. M. Welling	Universiteit van Amsterdam

Faculteit der Geesteswetenschappen

to Samira ...

Contents

Acknowledgments

It was wonderful!

I admit that I truly enjoyed it. I will miss these four years with all the great conversations, all the laughs, all the stress, all the deadlines, all the happiness and sadness after conference notifications, all the cool but never working ideas, all the not-so-great foods at the university cafeteria, all the foosball matches, all the soos talks and reading groups, all the slurm jobs, all the memory exhausted errors, all Chang's unfunny jokes, all the coffee breaks, and finally Bob.[1] But what I miss the most is the amazing group of people I met during these years.

In all these years I had the chance to **learn with perfect supervision** from Jaap and Maarten.

When I arrived in Amsterdam, Jaap was on a sailing trip with his boat (on which he never took me!) and at some point, someone told me he got stuck in a storm and I understood that my PhD under his supervision was going to be an adventurous journey. Jaap, thanks for giving me the opportunity in the first place, helping me to grow, teaching me how to do research, and chipping in with me for all the things that I wanted to do during my PhD.

I also wish to thank Maarten, who is an endless source of knowledge and experience (and knows a lot of good old songs!). Maarten has built ILPS, an excellent group that is a perfect platform to grow successful people. He cares for everyone in the group to have a great experience. It was an honor to be part of this group. Maarten, thanks for investing in me and for being a supportive supervisor and a fantastic mentor.

I am also honored to have Arianna, Bruce, Claudia, Evangelos, Max, and Stephan as my committee members. Thanks for generously offering your time.

The next one to thank is Hosein, my buddy. When we got the offer to do our PhD on the same project at the University of Amsterdam, I promised to play PES (Pro Evolution Soccer) with him every single day till the end of our PhD. Four years have passed and I still haven't played with him even a single time. My sincere apologies Hosein for not fulfilling my promise, but I will eventually make it up to you.

[1]Bob is the mammoth in the common room of third floor of Science Park 904.

The amazing people of ILPS and ILLC. Being part of these groups was the highlight of these four years.

I want to thank all the senior members and staff: Christof, Debbie, Evangelos, Fennike, Hinda, Jenny, Karine, Petra, Sonja, and Tanja. It was so pleasant to work with you. Special thanks to Maarten Marx for helping me at the beginning to get on board with my research. Maarten, thanks for all the fun and great conversations and remarkable guidance.

Also, I want to thank my fellow Master and PhD students, as well as great postdocs, for making these past years fun and exciting for me. Adith, Arezoo, Arnold, Aldo, Aleksandr, Alex, Alexey, Ali A, Ali V, Amir, Ana, Anna, Anne, Antonis, Arianna, Artem, Bob, Boris, Chang, Christophe, Chuan, Cristina, Daan, Damien, Dan, Dat, David, David, Dilek, Eva, Evgeny, Fei, Giorgos, Hamid, Harrie, Hendra, Hendrik, Hendrike, Hugo, Ilya, Isaac, Ivan, Jiahuan, Jie, Jin, Joost, Julia, Julien, Kaspar, Katya, Ke, Lars, Maartje, Mariya, Mahsa, Manos, Marc, Mariya, Marlies, Marzieh, Masrour, Maurits, Mohammad, Mozhdeh, Nikos, Olivier, Pengjie, Petra, Praveen, Richard, Ridho, Rolf, Sam, Sami, Shangsong, Shaojie, Spyretta, Svitlana, Tobias, Tom, Trond, Vera, Wanuy, Xiaohui, Xiaojuan, Xinyi, Yaser, Yangjun, Yifan, Zhaochun and Ziming, thanks for being so fantastic. I had so many great moments with every single one of you and I wish you all unlimited success and happiness.

During my PhD, I had the chance to intern at Google and Apple. I came back with plenty of new skills and ideas and warm memories of creative people there. Aleksey, Anna, Behrang, Enrique, Jakob, Jason, Lukasz, Manish, Marvin, Oriol, Pascal, Sascha, Stephan, Zoha, and Zina, thank you so much for making my internships such a great experience for me.

This is already getting really long, but I also want to thank my friends, colleagues, and family. They may not know how much they helped by just being in my life. Abbas, Afra, Ali, Alireza, Alexey, Amir, Amir Hossein, Amin, Arash, Arman, Aryaz, Atieh, Avital, Azadeh, Aziz, Babak, Behrooz, Casper, Danial, Dena, Elham, Erfan, Faegheh, Fahimeh, Fatemeh Zahra, Fereshteh, Hadi, Hamed, Hamid, Hamraz, Hoda, Hodjat, Hugo, Iman, Irene, Jonathan, Keyvan, Lasse, Leila, Liliana, Majid, Manoj, Maryam, Marzieh, Masoud, Masoumeh, Mehdi, Mehran, Melika, Mina, Mojdeh, Mojtaba, Mohammad, Mohammad Ali, Mohammad Hosein, Mohammad Reza, Mohammad Sadegh, Mohsen, Moslem, Nal, Nasrin, Neda, Niloofar, Noushin, Paniz, Panos, Parisa, Pegah, Pooya, Rasoul, Reza, Rianne, Rozhin, Sara, Sahar, Sanaz, Shayan, Shima, Spyretta, Thomas, Tim, Vahab, Yaser, and Zahra thank you so much. You are fantastic!

Mom, dad; this wouldn't be possible without you and your never-ending love and support. You are always my main driving motivation and I always admire you. Thanks for everything you have done for me. I wish I was able to spend more time with you during the past years, but you were in my heart every single moment.

I also want to thank the coolest siblings in the world, Mohsen and Zohre. Hanging out with you guys gave me a lot of positive energy during these years. You are the best!

Samira, however, tops the acknowledgments. Samira is the smartest and kindest person I know and having her next to me, every single day is just amazing. During these years, she listened to all my half-baked ideas and helped me solve many hard problems and fix so many nasty bugs, but more importantly, she adapted her life and decisions to just make it easier for me so many times and that is why I owe her a really giant thank you. Thank you for being so awesome, Samira!

Mostafa Dehghani
Winter of 2020.

"Give orange me give eat orange me eat orange give me eat orange give me you."

—Nim Chimpsky[2]

[2] Nim Chimpsky (November 19, 1973 – March 10, 2000) was a chimpanzee who was the subject of an extended study of animal language acquisition at Columbia University, led by Herbert S. Terrace, as a challenge to Chomsky's thesis that full-fledged language use was innate only to humans. This quote is the Nim's longest recorded sentence.

1 Introduction

Humans can learn about new concepts and solve complex problems from limited, noisy or inconsistent observations and routinely make successful generalizations based on them. Yet it remains a key challenge for machines to learn from such imperfect supervision. Most of today's successes of data-driven machine learning systems depend on the availability of massive amounts of high-quality labeled data.

In the context of human learning, the argument of learning with imperfect supervision has been advanced under the name *"poverty of the stimulus,"* proposed by Chomsky [48]. It suggests that human children can learn something as complex as a natural language from limited inputs of variable quality and need no evidence for much of the knowledge they bring to the learning process. To explain this, some researchers postulate the innateness of some knowledge. Following this line of thinking, Chomsky noted [48]:

> *"My own suspicion is that a central part of what we call "learning" is actually better understood as the growth of cognitive structures along an internally directed course under the triggering and partially shaping effect of the environment."*

Like humans, machines are confronted with the intriguing problem of the poverty of stimulus in learning and knowledge acquisition, although capturing such human-level learning abilities in machines, given imperfect supervision, remains a fundamental challenge in many domains [168]. In machine learning, a similar discussion has been put forward on how learning algorithms can deal with the problem of *poverty of stimulus*. It has been argued that a learner that makes no prior assumptions has no rational basis for generalizing over unseen instances [199], which implies "the futility of unbiased learning".[1]

The performance of machine learning models is often strongly correlated with the amount of available labeled data: the more data you have, the more accurate your model will be [121, 269]. However, in many real-world applications, large-scale high-quality training data is not available. This highlights the increasing need for building models with the ability to learn complex tasks with *imperfect supervision*. In this thesis, we use "*imperfect supervision*" as an umbrella term covering a variety of situations where the learning process is based on imperfect training samples [334].

We focus on language understanding and reasoning as a pivotal problem in artificial intelligence. Understanding language is one of the extraordinary cognitive abilities of humans. There have been several attempts to study whether non-human animals can acquire human language [219], like Project Nim,[2] a research project that was mounted in the 1970s to determine whether a chimpanzee, named Nim Chimpsky, raised in close contact with humans could develop a limited language. Regardless of whether at the end of the project Nim was using language to communicate or simply going through a bag of tricks to get things, Nim's language was much more limited compared to what a human child can develop in the early years. This indicates that achieving human-level understanding and generation of language would not be an easy goal for machines either and despite the good performance on specific benchmark datasets, machines are nowhere near the skill of humans at language understanding and reasoning.

Here, in this thesis, we concentrate on language understanding and reasoning in more principled ways to improve the learning process than ad-hoc and domain or task-specific tricks to improve the output. We investigate our ideas on a wide range of sequence modeling and language understanding tasks.

The main problem we study in this thesis is the poverty of stimulus for learning algorithms and how to overcome this problem, we focus on i) *employing prior knowledge*, ii) *augmenting data and learning to learn how to better use the data*, and iii) *introducing inductive biases into learning algorithms*, all to improve the learning process. We believe these approaches are important ways to move toward better machine learning systems that are able to generalize over observed imperfect supervision signals. As Mitchell [198] already states:

"If biases and initial knowledge are at the heart of the ability to generalize

[1]There has been a long discussion between rationalists and empiricists in philosophy and linguistics. Here, we just want to draw a connection between the process of learning in humans and machines in terms of the *poverty of stimulus*.

[2]*https://wikipedia.org/wiki/Nim_Chimpsky#Project_Nim*

beyond observed data, then efforts to study machine learning must focus on the combined use of prior knowledge, biases, and observation in guiding the learning process. It would be wise to make the biases and their use in controlling learning just as explicit as past research has made their observations and use."

There are many domains and applications that suffer from the lack of a large amount of high quality labeled data, since it maybe impossible or economically unreasonable to create such training datasets. Building machine learning systems that can learn with imperfect supervision dramatically extends the scope where AI-powered systems can be applied, and more generally, contributes to the broader goal of democratizing AI.

1.1 Problem Description and Research Questions

Curating a large amount of high quality labeled training data has become the primary bottleneck in developing new methods and applications in machine learning [226]. In practice, to deal with data scarcity in many tasks and applications, we can use higher-level approaches to provide supervision signals for training learning algorithms. This can be done for instance by:

- Using distant, or heuristic supervision [62, 79, 83, 85, 226, 226, 229, 251, 290];
- Using incidental signals that exist in the data and the environment independently of the tasks and that are co-related to the target tasks [239];
- Introducing a form of structured prior knowledge [66, 68];
- Exploiting noisy and inaccurate labels [31, 137, 172, 189, 216, 217, 226, 228, 284];
- Using indirect supervision, like providing supervision by specifying constraints that should hold over the output space [51, 265];
- Applying bootstrapping, self-supervised feature learning, and data augmentation to make statistically efficient reuse of available data [59, 94, 95, 202, 307];
- Using transfer learning to generalize knowledge across domains and tasks [241];
- Using active learning and response-based supervision in which the model receives feedback from interacting with an environment [51, 230];
- Zero/one/few-shot learning [99, 259, 260, 294];
- Injecting inductive biases into algorithms to generalize better on unobserved data [53, 54, 75].

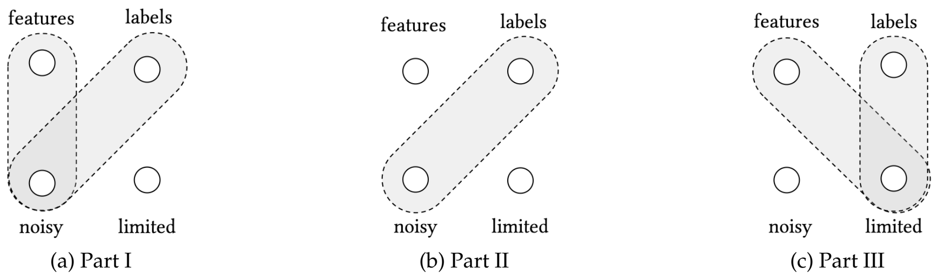


Figure 1.1: Different types of imperfection in the supervision that we deal with in each part of this thesis.

In a general supervised learning scenario, each training sample, which is used as the supervision signal, consists of a *feature vector* (also called data instance) and a *label*. However, in many domains and applications, there is an imperfection in the supervision signal. Here, in this thesis, we focus on two main problems: dealing with *noisy* training data and dealing with *limited* training data. We formulate the main research question addressed in this thesis as follows:

> **RQ-Main** *How can we improve the learning process for language understanding tasks, if the supervision signal is noisy in quality or limited in quantity?*

Based on the above terminology, i.e., *features* versus *label* and *noisy* versus *limited*, we can presume different types of imperfections in the supervision signal. In each part of this thesis, we target one or some of these types and proposed ideas that can improve the learning process with that type of imperfection in supervision. Figure 1.1 summarizes the different types of imperfection of the supervision that we consider in each part of this thesis. In Part I of the thesis, we address the problem of noisy features. We propose robust models that can deal with non-relevant terms in relevant documents when modeling the notion of relevance in the context of relevance feedback tasks for document ranking. We also propose models that are capable of detecting and ignoring unstable features that change over time when learning representations from data that evolves over time, as noisy factors in the data. We also partly address the problems of noisy labels by modeling the relevance in pseudo-relevance feedback tasks for document ranking, where top-k ranked documents in a retrieval run are assumed as relevant documents, while this assumption does not hold for all cases.

The proposed models in Part I organized around the idea of exploring how the structure of the data can be incorporated as prior knowledge to learn representations that are more robust against noise and changes in the data over time. The following research question is central to Part I of the thesis:

> **RQ-1** *How to use the structure of the data as prior knowledge to learn robust and effective representations of entities and concepts, when the data is noisy or variable over time?*

In Part II of the thesis, we target the problem of noisy or weak labels. We study how we can develop neural networks that can learn from weakly annotated training samples with the ability to go beyond the imperfection of the weak labels. We study different architectural choices and objective functions for neural ranking models to find a more noise-tolerant model when learning from pseudo-labels. We also introduce ideas that meta-learn the fidelity of weakly annotated labels and modulate the learning process based on the fidelity scores of weakly labeled samples. In fact, in this part, we make assumptions in the act of observing the data, like how to select the data based on the fidelity of data points, which is also a form of incorporating knowledge and biases. In this part, we address the following research question:

> **RQ-2** *How to design learning algorithms that can learn from weakly annotated samples, while generalizing over the imperfection in their labels?*

In Part III, we deal with the problem of limited labeled data, for instance, in the context of bAbI reasoning tasks with 1k training samples. We also study cases where we have a lot of labeled samples, but with limited coverage, which can be seen as a limitation in the diversity of feature vectors. For instance, in the context of algorithmic tasks, with the intention of assessing the ability of models on length generalization, we have plenty of training data but the distribution of sample's length in training is different from the test set. In this part, we investigate the idea of injecting some inductive biases[3] into models in order to encode modeling assumptions that help the models to be more data efficient and generalize better. The main research question that is addressed in Part III is:

> **RQ-3** *How can inductive biases help to improve the generalization and data efficiency of learning algorithms?*

1.1.1 Language Understanding and Sequence Modeling Tasks

We study our research questions in the context of sequence modeling and natural language understanding, generation, and reasoning tasks. In the course

[3]Inductive biases are "any biases for choosing one generalization over another, other than strict consistency with the observed training samples," as defined by Mitchell [198]. We elaborate on this definition at the beginning of Part III.

of this thesis, we cover a wide range of tasks, including:

- Assessing relevance for ranking (Chapter 2, 4, 5);
- (Pseudo)-relevance feedback for document ranking(Chapter 2);
- Contextual suggestion and recommendation [126] (Chapter 2);
- Text classification (over time) [139] (Chapter 3);
- Sentiment analysis [204, 237, 238] (Chapter 5);
- Machine Translation (Chapter 6);
- Natural language reasoning—bAbI tasks [302] (Chapter 6);
- Broad context language modeling [211] (Chapter 6);
- Open-domain question answering [90, 96] (Chapter 6);
- Modeling the structure in natural language sentences—subject-verb agreement task [178] (Chapter 6);
- Learning to execute computer programs [326] (Chapter 6);
- Algorithmic problems, like arithmetic and sequence memorization tasks [154] (Chapter 6).

Besides the challenges of learning from imperfect supervision in many of these tasks, some of them are difficult tasks that require, for instance, learning rich representations, or capturing complex underlying relations, or detecting and understanding abstract concepts and reasoning about them. Although we mainly focus on sequence modeling, text processing, and language-related tasks and evaluate our proposed ideas on these tasks, many of the proposed algorithms in this thesis can be easily extended to other domains like computer vision.

In the next section, we present an overview of the thesis and introduce the structure of the content and summarize different chapters in each part of the thesis.

1.2 Thesis Overview

This thesis consists of the introductory matter, followed by five main chapters that are divided into three parts as described in the previous section. [4] Finally, it closes off with the Conclusions and Bibliography.

PART I: Structure of the Data as Prior Knowledge

In this part, we explore how taking the general structure of data into account can help to estimate representations that capture only the significant features when the data is noisy and highly variant over time. We break our discussions in this part of the thesis into two chapters:

Chapter 2: Learning neither General, nor Specific, but Significant Representations

In this chapter, we address the following research question:

> **RQ-1.1** *How to learn robust representations for entities and abstract concepts that are affected by neither undiscerning general, nor noisy accidental features, given the structural relations in the data?*

We introduce *significant words language models* (SWLM) [62] to learn a representation for a set of textual entities, where this representation captures all, and only, the *significant* shared features from these entities. SWLM adjusts the weights of features to decrease the weight of noisy terms that are either well explained by all the entities, i.e., too general or only explained by a specific entity in the set, i.e., too specific, which eventually results in having the significant features left in the model. We employ SWLM in two language understanding tasks: the feedback problem in document ranking [64, 66], and group profiling in content personalization and recommendation tasks [67, 69]. We show how SWLM is remarkably robust against noisy features like non-relevant terms in relevant documents in the feedback task.

Chapter 3: Representational Separability for Hierarchically Structured Data

In this chapter, we address the following research question:

[4]This thesis consists of papers that contribute to the fields of information retrieval, natural language processing, and machine learning, thus the terminology used in different parts might slightly change accordingly.

> **RQ-1.2** *How to learn separable representations for hierarchically structured entities that are less sensitive to structural changes in the data and more transferable across time?*

We extend *significant words language models* to hierarchically structured data and introduce *hierarchical significant words language models* (HSWLM) [68, 70] which is an iterative approach that learns representations for hierarchical entities that are highly separable as SWLM removes the features that are well explained by either the ancestors (general features) or individual descendants (specific features). In this chapter, we discuss what makes separability a desirable property for classifiers and show how obtaining this property increases the robustness of representations against the structural changes in the data during the time.

PART II: Learning with Weak Supervision

In this part, we study how we can supervise machine learning systems by labeling training data programmatically instead of labeling by hand and discuss how to design neural networks that learn to go beyond the imperfection of the weakly annotated data. We break this into two chapters:

Chapter 4: Learning from Pseudo-Labels

In this chapter, we address the following research question:

> **RQ-2.1** *How can we train neural networks using programmatically generated pseudo-labels as a weak supervision signal, in a way that they exhibit superior generalization capabilities?*

We propose to train a neural ranking model using weak labels that are obtained automatically without human annotators or any external resources (e.g., click data). We train a set of simple yet effective neural ranking models and study their effectiveness under various learning scenarios, i.e., point-wise and pair-wise, different objective functions, and using different input representations [84]. We also discuss how privacy preserving approaches can benefit from models that are capable of learning from weak signals, where instead of labels from the original sensitive training data a noisy version is provided [71].

Chapter 5: Learning from Samples of Variable Quality

In this chapter, we focus on the following research question:

> **RQ-2.2** *Given a large set of weakly annotated samples and a small set of samples with high-quality labels, how can we best leverage the capacity of information in these sets to train a neural network?*

In this chapter we introduce *Learning with Controlled Weak Supervision (CWS)* and *Fidelity Weighted Learning (FWL)*, two semi-supervised approaches for training neural networks, where we have a large set of data with weak labels and a small amount of data with true labels. In CWS we train two neural networks in a meta-learning setup: a target network, the learner, and a confidence network, the meta-learner. The target network is optimized to perform a given task and is trained using a large set of unlabeled data that are weakly annotated. We propose to control the magnitude of the gradient updates to the target network using the scores provided by the second confidence network, which is trained on a small amount of supervised data. Thus we avoid that the weight updates computed from noisy labels harm the quality of the target network model.

FWL is a student-teacher approach in which we modulate the parameter updates to a *student* network (trained on the task we care about) on a per-sample basis according to the posterior confidence of its label-quality estimated by a *teacher* (who has access to the high-quality labels).

PART III: Injecting Inductive Biases for Data Efficiency

In this part, we discuss injecting inductive biases into learning algorithms as a way to help them to come up with more generalizable solutions when they are provided with limited observations. We further discuss how we can improve the generalization of Transformers [291], the self-attentive feed-forward networks for sequence modeling, by introducing a recurrent inductive bias into their architecture.

Chapter 6: Recurrent Inductive Bias for Transformers

In this chapter, we address the following research question:

> **RQ-3.1** *How can we improve the generalization and data efficiency of self-attentive feed-forward sequence models by injecting a recurrent inductive bias?*

We introduced the Universal Transformer [75], a self-attentive concurrent-recurrent sequence model, which is an extension of the Transformer model [292]. The Universal Transformer introduces recurrence in depth by repeatedly modifying a series of vector representations for each position of the sequence in parallel, by combining information from different positions using self-attention and applying a recurrent transition function across all time steps. In the simplest form, a Universal Transformer with a fixed number of iterations is almost equivalent to a multi-layer Transformer with tied parameters across all its layers. By sharing weights, we can save massively on the number of parameters that we are training and fewer parameters means that we learn faster with fewer data points. We show that the elegant idea of introducing recurrence in depth enables the Universal Transformer to extrapolate from training data much better on a range of algorithmic and language understanding tasks [72, 75].

While the three parts of the thesis offer a complete story aligned with the goal of *improving the learning process by incorporating prior knowledge, introducing the right biases, and making best use of the observations from the data*, they can be read independently. Each of the chapters of the thesis also has its own independent research questions, proposed ideas, and take home messages, but we recommend reading them in the order in which they appear in the parts.

1.3 Origins

Next, we present the origins of each chapter in terms of the papers they are based on.

Part I: *Structure of the Data as Prior Knowledge*

Chapter 2: *Learning neither General, nor Specific, but Significant Representations*

- Dehghani, M., Azarbonyad, H., Kamps, J., Hiemstra, D., and Marx, M. (2016c). Luhn revisited: Significant words language models. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16.
- Dehghani, M., Azarbonyad, H., Kamps, J., and Marx, M. (2016d). Generalized group profiling for content customization. In *CHIIR '16*, CHIIR '16.
- Dehghani, M., Azarbonyad, H., Kamps, J., and Marx, M. (2016f). Significant words language models for contextual suggestion. *Proceedings National Institute for Standards and Technology. NIST Special Publication: SP*, 500.
- Dehghani, M. (2016). Significant words representations of entities. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16 (SIGIR Doctoral Consortium Award).

MD designed the models, implemented the algorithms, ran the experiments, and did most of the writing. HA helped with the implementation and experiments. JK and DH helped with designing the model. HA, JK, DH, and MM contributed to the writing.

Chapter 3: *Representational Separability for Hierarchically Structured Data*

- Dehghani, M., Azarbonyad, H., Kamps, J., and Marx, M. (2016e). On horizontal and vertical separation in hierarchical text classification. In *The proceedings of ACM SIGIR International Conference on the Theory of Information Retrieval*, ICTIR'16 (Best Paper Award).
- Dehghani, M., Azarbonyad, H., Kamps, J., and Marx, M. (2016g). Two-way parsimonious classification models for evolving hierarchies. In *Proceedings of Conference and Labs of the Evaluation Forum*, CLEF '16 (Best Paper Honorable Mention).

MD designed the models, implemented the algorithms, ran the experiments, and did most of the writing. HA helped with the implementation and experiments. JK helped with designing the model. HA, JK, and MM contributed to the writing.

Part II: *Learning with Weak Supervision*

Chapter 4: *Learning from Pseudo-Labels*

- Dehghani, M., Zamani, H., Severyn, A., Kamps, J., and Croft, W. B. (2017g). Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17.
- Dehghani, M., Azarbonyad, H., Kamps, J., and de Rijke, M. (2017a). Share your model instead of your data: Privacy preserving mimic learning for ranking. In *SIGIR Workshop on Neural Information Retrieval*, SIGIR-NeuIR'17.
- Dehghani, M., Rothe, S., Alfonseca, E., and Fleury, P. (2017d). Learning to attend, copy, and generate for session-based query suggestion. In *Proceedings of The international Conference on Information and Knowledge Management*, CIKM'17.

MD designed the models, implemented the algorithms, ran the experiments, and did most of the writing. SR and HZ helped with designing the models. SR helped with the implementation. HZ, HA, AS, SR, and JK helped with designing the experiments. EA and PF mentored the (internship) project. HZ, HA, JK, BC, MdR, SR, AS, and PF helped with the writing.

Chapter 5: *Learning from Samples of Variable Quality*

- Dehghani, M., Mehrjou, A., Gouws, S., Kamps, J., and Schölkopf, B. (2018). Fidelity-weighted learning. In *International Conference on Learning Representations*, ICLR'18.

- Dehghani, M., Severyn, A., Rothe, S., and Kamps, J. (2017f). Learning to learn from weak supervision by full supervision. In *NIPS2017 workshop on Meta-Learning*, NIPS-MetaLearn'17.
- Dehghani, M., Severyn, A., Rothe, S., and Kamps, J. (2017e). Avoiding your teacher's mistakes: Training neural networks with controlled weak supervision. *arXiv preprint arXiv:1711.00313*.

MD designed the models, implemented the algorithms, ran the experiments, and did most of the writing. AM helped with designing the models. AM, SG, JK, and AS helped with designing the experiments. AM, SG, AS, SR, JK, and BS helped with the writing.

Part III: *Injecting Inductive Biases for Data Efficiency*

Chapter 6: *Recurrent Inductive Bias for Transformers*

- Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, Ł. (2019b). Universal transformers. In *International Conference on Learning Representations*, ICLR'19.
- Dehghani, M., Azarbonyad, H., Kamps, J., and de Rijke, M. (2019a). Learning to transform, combine, and reason in open-domain question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19.

MD co-designed the core model (first paper) with SG, and MD extended the model (second paper). MD implemented the algorithms, ran all the experiments, and helped with the writing. SG and OV mentored the (internship) project. SG led the writing. OV, JU, and LK helped with designing the model and designing the experiments. OV, JU, LK, HA, JK, and MdR helped with the writing.

The thesis also indirectly builds on the following papers (listed in reverse chronological order):

- Dehghani, M., Mehrjou, A., Gouws, S., Kamps, J., and Schölkopf, B. (2019c). Learning from samples of variable quality. In *ICLR workshop on Learning from Limited Labeled Data*, ICLR-LLD'19.

- Dehghani, M. (2018). Toward document understanding for information retrieval. *ACM SIGIR Forum*, 51(3).

- Zamani, H., Dehghani, M., Croft, W. B., Learned-Miller, E., and Kamps, J. (2018a). From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18.

- Azarbonyad, H., Dehghani, M., Kenter, T., Marx, M., Kamps, J., and de Rijke, M. (2018). HiTR: Hierarchical topic model re-estimation for measuring topical diversity of documents. *IEEE Transactions on Knowledge and Data Engineering*.

- Dehghani, M. and Kamps, J. (2018). Learning to rank from samples of variable quality. In *SIGIR 2018 Workshop on Learning from Limited or Noisy Data for Information Retrieval*.

- Zamani, H., Dehghani, M., Diaz, F., Li, H., and Craswell, N. (2018b). Workshop on learning from limited or noisy data for information retrieval. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '18.

- Azarbonyad, H., Dehghani, M., Marx, M., and Kamps, J. (2020). Learning to rank for multi label text classification: Combining different sources of information. In *Journal of Natural Language Engineering*.

- Azarbonyad, H., Dehghani, M., Beelen, K., Arkut, A., Marx, M., and Kamps, J. (2017a). Words are malleable: Computing semantic shifts in political and media discourse. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17.

- Kenter, T., Borisov, A., Van Gysel, C., Dehghani, M., de Rijke, M., and Mitra, B. (2017). Neural networks for information retrieval. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

- Dehghani, M., Jagfeld, G., Azarbonyad, H., Olieman, A., Kamps, J., and Marx, M. (2017b). On search powered navigation. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '17.

- Dehghani, M., Jagfeld, G., Azarbonyad, H., Olieman, A., Kamps, J., and Marx, M. (2017c). Telling how to narrow it down: Browsing path recommendation for exploratory search. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*, CHIIR '17.

- Dehghani, M., Jagfeld, G., Azarbonyad, H., Olieman, A., Kamps, J., and Marx, M. (2017c). Telling how to narrow it down: Browsing path recommendation for exploratory search. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*, CHIIR '17.

- Azarbonyad, H., Dehghani, M., Kenter, T., Marx, M., Kamps, J., and de Rijke, M. (2017b). Hierarchical re-estimation of topic models for measuring topical diversity. In *European Conference on Information Retrieval (ECIR'17)*.

- Dehghani, M., Azarbonyad, H., Kamps, J., Hiemstra, D., and Marx, M. (2016b). Inoculating relevance feedback against poison pills. In *15th Dutch-Belgian Information Retrieval Workshop, DIR 2016*.
- Dehghani, M., Abnar, S., and Kamps, J. (2016a). The healing power of poison: Helpful non-relevant documents in feedback. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16.
- Quiroz, L., Mennes, L., Dehghani, M., Kanoulas, E., et al. (2016). Distributional semantics for medical information extraction. In *CLEF Working Notes*, pages 109–122.
- Hashemi, S. H., Dehghani, M., and Kamps, J. (2015a). Parsimonious user and group profiling in venue recommendation. *Proceedings National Institute for Standards and Technology*, 500.
- Dehghani, M., Azarbonyad, H., Marx, M., and Kamps, J. (2015a). Learning to combine sources of evidence for indexing political texts. In *Proceedings of the Dutch-Belgian Information Retrieval Workshop*.
- Tabrizi, S. A., Dadashkarimi, J., Dehghani, M., Nasr Esfahani, H., and Shakery, A. (2015). Revisiting optimal rank aggregation: A dynamic programming approach. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, ICTIR '15.
- Azarbonyad, H., Saan, F., Dehghani, M., Marx, M., and Kamps, J. (2015b). Are topically diverse documents also interesting? In *International Conference of the Cross-Language Evaluation Forum for European Languages (CLEF)*.
- Azarbonyad, H., Dehghani, M., Marx, M., and Kamps, J. (2015a). Time-aware authorship attribution for short text streams. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15.
- Dehghani, M., Azarbonyad, H., Marx, M., and Kamps, J. (2015b). Sources of evidence for automatic indexing of political texts. In *Proceedings of European Conference on IR Research*, ECIR'15.
- Olieman, A., Azarbonyad, H., Dehghani, M., Kamps, J., and Marx, M. (2014). Entity linking by focusing dbpedia candidate entities. In *Proceedings of the First International Workshop on Entity Recognition and Disambiguation*, ERD '14

Part I

Structure of the Data as Prior Knowledge

The world is structured or at least, humans understand it in structural terms. We approach new problems armed with extensive *prior experience and knowledge* about these structures. When we learn, we either fit the new knowledge into our existing structured representations or we adjust the existing structure to better accommodate our new and the old observations [22]. Our brain uses *hierarchies* to abstract away fine-grained differences and model different levels of associations [21]. Using such a structure as a prior, we automatically condense the information, which not only has the effect of removing impurity of the data but also eliminates unnecessary details, which eventually helps us to memorize, think and reason in a more efficient way.

When building intelligent machines, we need to consider that real-world data can be complex, limited, highly variable, and noisy. However, in many cases, there is a general structure that the data follows and taking this structure into account facilitates modeling complex information, adds biases to compensate for the limited availability of the data, discovers robust knowledge from the data, and learns representations that are less affected by non-essential features. In Part I of this thesis, we address one of our research questions:

> **RQ-1** *How to use the structure of the data as prior knowledge to learn robust and effective representations of entities and concepts, when the data is noisy or variable over time?*

Understanding language is a complex task and given imperfect supervision, considering no knowledge as a prior in the learning process seems to be unrealistic even for humans:

> *"Narrowly limited extent of the available data [...] leaves little hope that much of the structure of the language can be learned by an organism initially uninformed as to its general character."* Chomsky [46]

In this part of the thesis, we focus on transforming textual data into digestible representations, which is the key step for any data-oriented method employed for language understanding [23]. We are concerned with questions surrounding how we can best learn representations that are *precise*, *robust against noise*, and *transferable over time* for textual entities that can be *abstract concepts for which we have no direct observation*, for instance, learning to model topical relevance, given a set of documents that are labeled as relevant, like learning a representation for a parliamentary party, given speeches of its member.

In the first chapter of this part, Chapter 2, we address the following research question:

> **RQ-1.1** *How to learn robust representations for entities and abstract concepts that are affected by neither undiscerning general, nor noisy accidental features, given the structural relations in the data?*

We introduce *significant words language models* (SWLM) [62] of a set of documents, that capture all, and only, the significant shared terms from these documents. This is achieved by adjusting the weights of terms already well explained by the document collection as well as the weight of terms that are only explained by individual documents, which eventually results in having the significant terms left in the model. We apply the resulting models to two main language understanding applications: the feedback problem in information retrieval [64, 66], and group profiling in content personalization and recommendation tasks [67, 69]. We further show that SWLM are remarkably robust representations that are insensitive to the noisy terms and at the same time interpretable by human inspection.

Then, in the second chapter of this part, Chapter 3, we get down to the following research question:

> **RQ-1.2** *How to learn separable representations for hierarchically structured entities that are less sensitive to structural changes in the data and more transferable across time?*

We extend *significant words language models* to hierarchical structures and introduce *hierarchical significant words language models* (HSWLM) [68, 70] that learn representations for hierarchical entities. HSWLM iteratively sparsifies the representation of the entities by discarding features that are well explained by their ancestors, i.e., general features, as well as features that reflect the characteristics of individual descendants, i.e., specific features. This leads to representations for entities that are both vertically and horizontally separable, in terms of their position in the hierarchy. We discuss what makes separability a desirable property for classifiers and show how obtaining this property leads to time-agnostic representations, i.e., representations that are invariant to the structural changes in the data during the time.

We show that by taking the structure and relations in the data as a prior knowledge into account, we can effectively learn representations that are less affected by noisy variant factors in the data. This can be also consider as a particular form of inductive bias that we will further discuss in Part III. We use the learned representations to solve different language understanding tasks and observe boosts in performance, in particular, due to the robustness of the learned representations against noise.

2 Learning neither General, nor Specific, but Significant Representations

When learning representations of concepts and entities from the data, there are general features that are propertyless common observations that do not have enough expressivity to make distinctions, and specific features that are unreliable rare observations that do not generalize to all instances. Taking the structure and the relations in the data into account can help learning representations that capture only and all significant features and are less affected by the noisy factors in the data.

2.1 Introduction

Humans instinctively know how to efficiently "select details" that are useful for recognition and at the same time "abstract away" unnecessary or noisy details [22, 107, 276]. This is, in fact, a survival trait, since we cannot possibly save all the detail around us in our brain [278]. Hence, to define objects and actions, we can extract a subset from an extensive but finite set of existing features that are neither *too specific*, nor *too general*, but *significant*. In many cases, this is done by analyzing inheritance (is-a) and composition (has-a) relationships

This chapter is based on [62, 66, 67, 69].

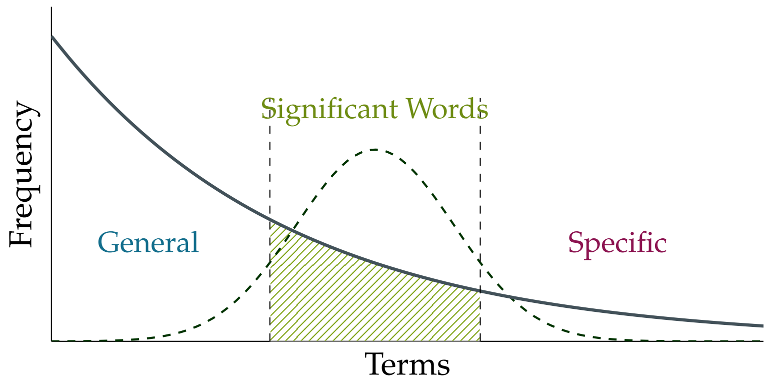


Figure 2.1: Establishing a set of "Significant Words" based on Luhn [183].

between concepts, entities, and actions [28, 110].

Inspired by the way that human process the information at different levels of abstraction, in this chapter, we explore ideas to address one of our research questions:

> **RQ-1.1** *How to learn robust representations for entities and abstract concepts that are affected by neither undiscerning general, nor noisy accidental features, given the structural relations in the data?*

We present our ideas and discussions in the context of language understanding tasks, in particular learning representations for textual entities for information retrieval applications. As the modeling approach, we estimate a unigram language model, or a so-called "bag of words" model, to represent an entity, like a person, organization, category, etc, given a set of texts connected to the entity. In fact, we assume that the textual documents associated with an entity are samples drawn from the distribution of a model that represents the entity and then estimate the model given these samples.

More specifically, we introduce *significant words language models* (SWLM) [66, 67, 69] as a family of models aimed at learning representations for an entity, given a set of documents associated with the entity, so that *all, and only*, the significant shared terms are captured in the representations. As a result, representations learned by these models are not only distinctive but also supported by all the documents in the set. In short, this is achieved by adjusting the weights of terms already well explained by all the existing documents in the collection as well as the weights of terms that are only explained by a specific document in the set, which eventually results in having the significant terms left in the model.

The general idea of SWLM is inspired by the early work of Luhn [183], in which he argues that to extract **significant words**, we need to avoid both common observations and rare observations. Luhn assumed that frequency data can be used to measure the significance of words concerning their ability to represent a piece of text. Considering Zipf's Law, he simply devised a counting technique for finding significant words where he specified two cut-offs based on collection frequency of terms, an upper and lower (see Figure 2.1), to exclude non-significant words.

There have been efforts to bring this idea to estimate a more precise language model, like mixture models [329] and parsimonious language models [136]. These work tried to improve the raw language model by eliminating the effect of common terms from the model. However, instead of using fixed frequency cut-offs, they made use of a more advanced way to do this. Hiemstra et al. stated the following in their paper:

> *[...] our approach bears some resemblance with early work on information retrieval by Luhn, who specifies two word frequency cut-offs, an upper and a lower to exclude non-significant words. The words exceeding the upper cut-off are considered to be common and those below the lower cut-off rare, and therefore not contributing significantly to the content of the document. Unlike Luhn, we do not exclude rare words and we do not have simple frequency cut-offs [...]*

In a way, the idea of SWLM completes the cycle, implementing the vision of Luhn. We introduce a meaningful translation of both *specificity* and *generality* against *significance* in the context of learning representation for an entity given the set documents associated with it. Then we propose an effective way of estimating a model in which we learn a distribution over terms that is affected by neither the common observations nor the rare observations.

While estimating SWLM, as a distribution over terms to represent an entity given the set of documents associated with it, we cast aside terms that are not specific enough to reflect features of the entity that makes its representation distinguishable from other entities. At the same time, we abstract away from noisy factors of variation that are document specific terms that are not general enough to describe all the documents as a set representing the entity.

2.1.1 Preliminaries

In Part I of this thesis, i.e., Chapter 2 and 3, an "entity" can refer to a concept, a person, an organization, a category, or an ideology [62]. The entity can be an abstract entity for which we have *no direct observation*, for instance, learning a representation for a parliamentary party, given speeches of its member, or learning to model relevance, given a set of documents that are labeled as relevant. We assume the cases where the entity is associated with a set of textual documents. The textual documents associated with an entity, for instance, for the aforementioned examples, can be speeches given by the person, the documents published by the organization, the associated text by instances of the category, and the set of documents describing the ideology. Here, as the modeling approach, we use "language model" and we stick to the simplest form of language models, unigram language model, in which we assume that a word sequence is generated by generating each word independently that specifies a multinomial distribution over all the words. Thus, the probability of a sequence of words would be equal to the product of the probability of each word.

In order to model an observed sequence of words d, we assume it is generated using a unigram language model θ and we would like to infer θ, i.e., estimate the probability of each word w given the model, $p(w|\theta)$, based on the observed d.

As the standard and simple of estimating the language model, we can use maximum likelihood estimator and find the $\hat{\theta}$ that gives the observed data the highest likelihood:

$$\hat{\theta} = \arg\max_{\theta} p(d|\theta). \tag{2.1}$$

By writing down the log-likelihood function and using the Lagrange multiplier to combine the constraint of $\sigma_{winv} p(w|d) = 1$ with the original log-likelihood function, we can get to a new unconstrained optimization problem. Then, by taking partial derivatives of this function and setting them to zero, we can obtain the solution for θ which gives each word a probability equal to its relative frequency in d:

$$p(w|\hat{\theta} = \frac{c(w,d)}{|d|}, \tag{2.2}$$

where $c(w,d)$ is the count of word w in d and $|d|$ is the length of d that is equal to total number of word occurrences in d.

The unigram language model clearly makes unrealistic assumptions about independent word occurrences in texts. To address this limitation, N-gram language model captures some limited dependency between words and assumes the occurrence of a word depends on the proceeding $n-1$ words. However, as the complexity of a language model increases, so does the number of parameters and we would need much more data to infer the model. Moreover, the computational cost of complex language models is also a concern for some of the large-scale retrieval or classification tasks. In information retrieval, bigram or trigram language models tend not to improve much over the unigram language model. This might be due to the data sparseness where bigram or trigram language models overfit, or the information about presence or absence of words and word frequencies may be sufficient to determine the relevance of a document while the exact word order may not be so important, unlike other language understanding tasks like machine translation or speech recognition. For machine translation, unigram language models are clearly insufficient and more sophisticated language models would be needed [328].

Since a language model is a probabilistic model of text data, a direct way of evaluating a language model would be to assess how well the model fits the test data, for instance, using the likelihood of the test data given a model to be evaluated, where a higher likelihood would indicate a better fit, thus a better language model. However, fitting the data well does not necessarily imply better performance for the task at hand. To avoid this gap, here we use an indirect way of evaluating the quality of a language model by assessing their contribution to the retrieval or classification performance.

2.1.2 Detailed Research Questions

We break down our main research question in this chapter into three concrete research questions:

> **RQ-1.1.1** *How to estimate a representation for a set of entities that captures all, and only, the essential shared commonalities of these entities?*
>
> **RQ-1.1.2** *How do significant words language models capture the mutual notion of relevance for a set of feedback documents and prevent noisy terms by controlling the contribution of each of the documents in the feedback model?*
>
> **RQ-1.1.3** *How well can significant words language models profile groups of entities and how effective are these profiles in content customization tasks?*

In the following sections, we will address these research questions one by one.

2.2 Significant Words Language Models

In this section, we address the first research question of this chapter:

> **RQ-1.1.1** *How to estimate a representation for a set of entities that captures all, and only, the essential shared commonalities of these entities?*

We introduce Significant Words Language Models and describe how to estimate them. SWLM assumes that terms in the associated documents are drawn from three models: 1. A *general model*, representing common observations, 2. A *specific model*, representing partial observations, and 3. A *significant words model*.

The significant words model is the latent model representing the essential features of the entity. The general and specific models, however, are not necessarily topic-centric models. In a way, they are supposed to represent two distributions of terms that are not considered as significant information.

Each model is represented using a terms distribution, i.e., a unigram language model, θ_{sw}, θ_g, and θ_s. We assume that each term in a document in the set is generated by sampling from a mixture of these three models independently. Thus, the probability of appearance of the term t in the document d is as follows:

$$p(t|d) = \lambda_{d,sw} p(t|\theta_{sw}) + \lambda_{d,g} p(t|\theta_g) + \lambda_{d,s} p(t|\theta_s), \tag{2.3}$$

where $\lambda_{d,x}$ stands for $p(\theta_x|d)$ which is the probability of choosing the model θ_x given the document d.

We estimate the general and specific models based on patterns of occurrences of terms in different documents in the set and fix them in the estimation process as infinitely strong priors. We consider the collection model, i.e., the set of all documents, θ_C as an estimation for θ_g:

$$p(t|\theta_g) = p(t|\theta_C) = \frac{c(t,C)}{\sum_{t' \in V} c(t',C)}, \tag{2.4}$$

where $c(t,C)$ is the frequency of term t in the collection. This way, terms that are well explained by the collection model get high probability and are considered as general terms.

Furthermore, we establish a definition for "specificity" with regards to our main goal, which is estimating a representation for a set of documents, as being supported by part of the documents in the set but not all. We estimate θ_s to represent the probability of a term being partially observed as follows, and normalize all the probabilities to form a distribution:

$$p(t|\theta_s) = \sum_{d_i \in \mathcal{D}} \left(p(t|\theta_{d_i}) \prod_{\substack{d_j \in \mathcal{D} \\ j \neq i}} (1 - p(t|\theta_{d_j})) \right), \tag{2.5}$$

where $P(t|\theta_{d_i}) = {}^{c(t,d_i)}/_{\sum_{t' \in d_i} c(t',d_i)}$. Intuitively, Equation 2.5 defines the probability of term t to be a *specific* term, based on how much this term is important in one of the documents but not others, marginalizing over all the documents in the set D. This way, terms that are well explained in only one document but not others get higher probabilities and are considered as insignificant specific terms.

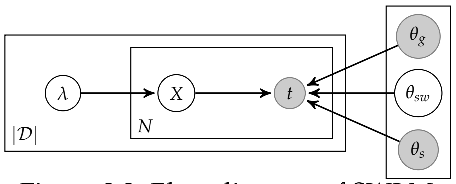


Figure 2.2: Plate diagram of SWLM.

Having the above assumptions, the goal is to fit a log-likelihood model of generating all terms in the documents in the set to discover the term distribution of the significant words model, θ_{sw}. Let $\mathcal{D} = \{d_1, \ldots, d_{\mathcal{D}}\}$ be the set of documents associated withe the entity we want to learn a representation for. The log-likelihood function for the entire set of documents is:

$$\log p(\mathcal{D}|\mathrm{Y}) = \sum_{d \in \mathcal{D}} \sum_{t \in V} c(t,d) \log \left(\sum_{x \in \{sw,g,s\}} \lambda_{d,x} p(t|\theta_x) \right), \tag{2.6}$$

where $c(t,d)$ is the frequency of the term t in the document d, and Y determines the set of all parameters that should be estimated, $\mathrm{Y} = \{\lambda_{d,sw}, \lambda_{d,g}, \lambda_{d,s}\}_{d \in \mathcal{D}} \cup \{\theta_{sw}\}$.

To fit our model, we estimate the parameters using the maximum likelihood (ML) estimator. Therefore, assuming that documents are represented by a multinomial distribution over the terms, we solve the following problem:

$$\mathrm{Y}^* = \arg\max_{\mathrm{Y}} p(\mathcal{D}|\mathrm{Y}) \tag{2.7}$$

Assuming that $X_{d,t} = \{sw, g, s\}$ is a hidden variable indicating which model has been used to generate the term t in the document d, we can compute the parameters using the Expectation-Maximization (EM) algorithm. The stages of the EM algorithm are as follows:

E-Step

$$p(X_{d,t} = x) = \frac{p(\theta_x|d)p(t|\theta_x)}{\sum_{x' \in \{sw,g,s\}} p(\theta_{x'}|d)p(t|\theta_{x'})} \tag{2.8}$$

M-Step

$$p(t|\theta_{sw}) = \frac{\sum_{d \in \mathcal{D}} c(t,d)p(X_{d,t} = r)}{\sum_{t' \in V} \sum_{d \in \mathcal{D}} c(t',d)p(X_{d,t'} = r)} \tag{2.9}$$

$$\lambda_{d,x} = p(\theta_x|d) = \frac{\sum_{t \in V} c(t,d)p(X_{d,t} = x)}{\sum_{x' \in \{sw,g,s\}} \sum_{t \in V} c(t,d)p(X_{d,t} = x')} \tag{2.10}$$

Figure 2.2 represents the plate notation of SWLM. As it is shown, for each document the contribution of each of the three models, i.e., $\lambda_{d,x}$ for $x \in \{sw, g, s\}$, are estimated. It can be seen that the general model, θ_g, and the specific model, θ_s are considered as external observations, which are involved in the estimation process as infinitely strong priors.

In the next section, we employ SWLM in different applications in order to assess its effectiveness in modeling an entity given a set of documents that are associated with the entity. We evaluate SWLM on relevance feedback in the retrieval task, where we need to model the set of relevant documents or top-ranked documents in the initial retrieved results and use this model to expand the user's query to improve the retrieval performance by shrinking the vocabulary gap between query and relevant documents. Furthermore, we use SWLM as a group profiling approach in the task of contextual suggestion to model preferences of a group of people and augment user profiles with the profiles of implicit or explicit groups they belong to.

2.3 SWLM for Relevance Feedback

Modeling and assessing "relevance" is an important goal in information retrieval and search that requires understanding users' queries, documents, and the relation between them. One of the key factors affecting search quality is the fact that user queries are ultra-short statements of their complex information needs. Query expansion has been proven to be an effective technique to bring agreement between user information need and relevant documents [125]. Taking feedback information into account is a common approach for enriching the representation of queries and consequently improving retrieval performance.

In True Relevance Feedback (TRF), given a query and a set of judged documents, either explicitly assessed by the user or implicitly inferred from user behavior, the system tries to enrich the query representation to improve the performance of the retrieval. However, feedback information is not available in most practical settings. An alternate approach is Pseudo Relevance Feedback (PRF), also called blind relevance feedback, which uses the top ranked documents in the initial retrieved results for the feedback.

The main goal of feedback systems in the retrieval task is to make use of feedback documents to estimate a more accurate query model representing the notion of relevance. However, although documents in the feedback set contain relevant information, there is always also non-relevant information. For instance, in PRF, some documents in the feedback set might be non-relevant, or in TRF, some documents, despite the fact that they are relevant, may contain off-topic information and act like poison pills [277] by hurting the performance of feedback systems. Such non-relevant information can distract the feedback model by adding bad expansion terms, leading to *topic drift* [130, 188]. It has

Standard-LM	
prize	5.55e-02
nobel	3.36e-02
physics	2.35e-02
science	2.18e-02
...	
time	1.68e-02
...	
palestinian	1.34e-02
year	1.34e-02
...	

General-LM	
new	3.70e-03
cent	2.98e-03
two	2.97e-03
dollars	2.76e-03
people	2.71e-03
...	
time	2.47e-03
...	
year	2.16e-03
...	

SMM [329]	
prize	6.07e-02
nobel	4.37e-02
awards	3.43e-02
chemistry	3.23e-02
physics	2.82e-02
palestinian	2.18e-02
cesium	2.09e-02
arafat	1.94e-02
university	1.92e-02
...	

Specific-LM	
insulin	2.25e-02
palestinian	2.15e-02
dehmelt	1.81e-02
oscillations	1.79e-02
waxman	1.69e-02
marcus	1.69e-02
attack	1.61e-02
...	
arafat	1.29e-02
...	

SWLM	
prize	6.02e-02
nobel	4.53e-02
science	2.68e-02
award	2.43e-02
physics	1.94e-02
winner	1.90e-02
won	1.80e-02
peace	1.80e-02
discovery	1.71e-02
...	

Figure 2.3: Extracting *significant terms* from relevant feedback documents. (Topic 374 of the TREC Robust04 test collection: "Nobel prize winners".)

been shown that based on this observation, existing feedback systems are able to improve the performance of the retrieval if feedback documents are not only relevant, but also have a dedicated interest in the topic [130].

Significant words language model seems a great fit to this situation as it models the the set of feedback documents by capturing the essential terms representing a *mutual notion of relevance*, i.e., a representation of characteristic terms which are supported by all the feedback documents.

Figure 2.3 shows an example of estimating language models from the set of top seven relevant documents retrieved for topic 374, "Nobel prize winners," of the TREC Robust04 test collection. Terms in each list are selected from the top-50 terms of the models estimated after stop word removal. Standard-LM is the language model estimated using MLE considering feedback documents as a single document. SMM is the language model estimated using the mixture model [329], one of the most powerful feedback approaches, which generally tries to remove background terms from the feedback model.

General-LM denotes the probability of terms to be common based on their overall occurrence in the collection and Specific-LM determines the probability of terms to be specific in the feedback set, i.e., being frequent in one of the feedback documents but not others. The way in which the General-LM and Specific-LM are estimated has been discussed in detail in Section 2.2. And the last model in the figure is SWLM, which is the extracted latent model with regards to General-LM and Specific-LM, using significant words language models.

As can be seen, by considering feedback documents as a mixture of feedback model and collection model, the mixture model [329] penalizes some general terms like "time" and "year" by decreasing their probabilities. However, since some frequent words in the feedback set are not frequent in the whole collection, their probabilities are boosted, like "Palestinian" and "Arafat," while they are not good indicators for the whole feedback set. The point is that although these terms are frequently observed, they only occur in some feedback documents not most of them, which means that they are in fact "specific" terms, not significant terms. By estimating both a general model and a specific model and taking them into consideration, SWLM tries to control the contribution of each feedback document in the feedback model, based on its merit, and prevent the estimated model to be affected by indistinct or off-topic terms, resulting in a significant model that reflects the notion of relevance.

Here, we are going to address our second research question of this chapter:

> **RQ-1.1.2** *How do significant words language models capture the mutual notion of relevance for a set of feedback documents and prevent noisy terms by controlling the contribution of each of the documents in the feedback model?*

We explain how to estimate SWLM for a set of feedback documents and discuss its effectiveness in this task along with a set of analyses and ablation studies.

2.3.1 Language Models for Feedback

In information retrieval, language modeling is usually used to represent the user query by estimating a query language model, θ_q, based on maximum likelihood estimation: $p(t|\theta_q) = c(t,q)/|q|$, where $c(t,q)$ is the frequency of term t in q and $|q|$ is the total number of terms in the query. Then, after applying smoothing methods on the language model of documents [330], the KL-divergence is employed to score documents based on the negative KL-divergence between the estimated language models of the query and each document document [167]:

$$Score(d,q) = -D(\theta_q||\theta_d). \tag{2.11}$$

In the retrieval tasks, there might be a lack of agreement between the user and the system in the form of the vocabulary missmatch between the user's query and relevant documents. To address this problem, a feedback language model, $\theta_{\mathcal{F}}$ that is estimated given the set of feedback documents is used to expand the user's query. A common approach for expanding the query is interpolating $\theta_{\mathcal{F}}$ with the original query model [2, 329]:

$$p(t|\theta'_q) = (1-\alpha)p(t|\theta_q) + \alpha p(t|\theta_{\mathcal{F}}), \tag{2.12}$$

where α controls the amount of feedback. Thereafter the expanded query model is used in Equation 2.11 for ranking the documents.

The main goal of feedback methods is to estimate an effective feedback model, $\theta_{\mathcal{F}}$, from the set of feedback documents. We can use significant words language models to represent feedback documents. In this context, the significant words are in fact words that are reflecting the notion of relevance. We use the approach described in Section 2.2, where we take the θ_{sw} as the $\theta_{\mathcal{F}}$ and use it in Equation 2.12 for expanding the query.

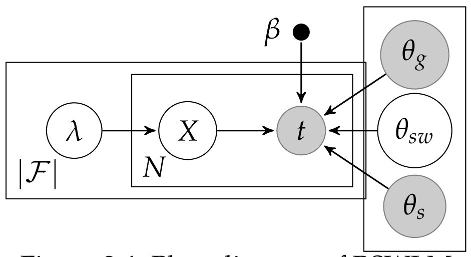


Figure 2.4: Plate diagram of RSWLM.

2.3.2 Regularized SWLM

Using the original significant words language models for estimating the feedback model, the original query model has not been considered for estimating the feedback model. Thus, in case where we only have a few relevant documents in the feedback set for a query, the model could be distracted by non-relevant information and converge to a local optimum point.

To cope with this problem, and avoid degradation in the performance, a solution is to involve information from the original query [124]. Inspired by the work by Tao and Zhai [274], we modify the estimation process of SWLM and estimate Regularized Significant Words Language Models (RSWLM) by incorporating the extra knowledge from the query model. We define a prior parameter and employ maximum a posteriori to fit the model to feedback documents and solve the following problem:

$$\mathrm{Y}^* := \arg\max_{\mathrm{Y}} p(\mathcal{F}|\mathrm{Y})P(\mathrm{Y}) \tag{2.13}$$

We define the a conjugate Dirichlet prior on θ_{sw} as follows:

$$p(\theta_{sw}) \propto \prod_{t \in V} p(t|\theta_{sw})^{\beta p(t|\theta_q)}, \tag{2.14}$$

where $\beta p(t|\theta_q)$ is the parameter of the Dirichlet distribution which in fact performs as the additional pseudo-count for t to push the model θ_{sw} to assign a higher probability to term t as it has a high probability in θ_q.

Generally speaking, this adds a bias in the estimation process to bend the feedback model toward the original query model. Here, the value of β controls the amount of this bias. Taking the conjugate prior into account, we conduct the MAP estimation by updating Equation 2.9 in the EM algorithm as follows:

$$p(t|\theta_{sw}) = \frac{\sum_{d \in \mathcal{F}} c(t,d) p(X_{d,t} = sw) + \beta p(t|\theta_q)}{\sum_{t' \in V} \sum_{d \in \mathcal{F}} c(t',d) p(X_{d,t'} = sw) + \beta} \tag{2.15}$$

So, by modifying the EM algorithm, we consider our observation from the query model as a pseudo-document which makes the feedback model become more rigid.

Table 2.1: Statistics of the collections used for experimental evaluations.

Dataset	TREC Track	Queries	#Docs	#Queries in TRF exp.
Robust04	TREC'04 Robust	301–450 and 601–700	528,155	217
WT10G	TREC'09–'10, Web ad-hoc	451-550	1,692,096	81
GOV2	TREC'04–'06 Terabyte Track	701-850	25,178,548	134

Similar to the approach in [274], we initialize β with a large value, and then dynamically decrease its value in each EM iteration until the point that we have equal contributions of the original query and the feedback documents.

Figure 2.4 represents the plate notation of regularized significant words language models. For each document contributions of each of the three models, i.e., λ's, are estimated. The general model, θ_g, and the specific model, θ_s are considered as external observations, which are involved in the estimation process as infinitely strong priors. It is noteworthy that fixing these parameters also helps to decrease the number of local maximums. As illustrated in the diagram, β plays the role of regularizing parameter.

Establishing a model consisting only of significant words using the fixed cutoffs based on frequency of terms, as was originally proposed by Luhn [183], runs the risk of leaving good expansion terms out, especially trimming the model toward specific terms may lead to the loss of discriminative relevant terms that can have a high impact on retrieval effectiveness. SWLM enables us to reduce this risk as estimating a specific language model using Equation 2.5, which empowers our estimation process to retain the significant terms that are globally infrequent, but well supported by the feedback documents.

2.3.3 Experimental Setup

In this section, we describe the test collections used in our experiments as well as the settings of our experiments. We use the Robust04[1], WT10G[2], and GOV2[3] test collections, which are different in terms of both size and genre of documents. Information about each collection is summarized in Table 2.1.

We have employed the Lemur toolkit and Indri[4] search engine to carry out our experiments. We have implemented SWLM and RSWLM in the Lemur project framework. In all our experiments, we only use the "title" field of the TREC topics as queries. We have used the Porter stemmer for stemming all queries and document's terms and removed stopwords using the standard InQuery

[1]*https://trec.nist.gov/data/robust/04.guidelines.html*

[2]*http://ir.dcs.gla.ac.uk/test_collections/wt10g.html*

[3]*http://ir.dcs.gla.ac.uk/test_collections/gov2-summary.htm*

[4]*http://www.lemurproject.org/*

stopword list. We have used the KL-Divergence model [167], with Dirichlet smoothing [330], as the retrieval model in all of the experiments, including initial retrieval as well as feedback runs. We set the Dirichlet smoothing prior to $1,000$. In the feedback runs, for each collection and each method, we have performed a full grid search and tuned the three main parameters (the value of the feedback interpolation coefficient, the number of feedback documents, and the number of expansion terms) by dividing the queries into three folds and conducting 3-fold cross-validation with the same split for folding in all the experiments. Also we have tuned the hyperparameters of each method during the cross validation.

The Mean Average Precision (MAP) performance measure for top-$1,000$ documents is used as the evaluation metric. Moreover, we report P@10 (precision at 10) for PRF and P@20 (precision at 20) for TRF as the indicators of the precision for the *first* result page and *first-two* result pages, respectively. To avoid the ranking effect[5] [50] in the evaluation of the TRF task, we have used the modified freezing technique[6] in the evaluation of the results of these experiments [124, 243]. In addition to the above metrics, we also report robustness index, $RI(Q)$, which is also called reliability of improvement [55]. For a set of queries Q, the RI measure is defined as: $RI(Q) = {}^{N^+ - N^-}/_{|Q|}$, where N^+ is the number of queries helped by the feedback method and N^- is the number of queries hurt.

In our experiments, as the baseline methods, we have used the most popular unsupervised state-of-the-art methods for the feedback task that are proposed in the language modeling framework. Our baseline methods are: the maximum likelihood estimation—without feedback (MLE) [167], the simple mixture model (SMM) [329], the divergence minimization model (DMM) [329], the relevance models (RM3 and RM4) [2, 169], the regularized mixture model (RMM) [274], and the maximum-entropy divergence minimization model (MEDMM) [187].

2.3.4 SWLM for Relevance Feedback

Now, we present the results of applying SWLM on both true and pseudo relevance feedback tasks.

[5]In TRF, since relevant documents already seen by the user are usually moved to the top of the ranking, thereby distorting the feedback evaluation, making it seem really good, while most of the improvement is gained simply by a reranking of documents already seen. This is known as "ranking effec".

[6]Modified freezing is a technique to eliminate the "ranking effect" and evaluate only the "feedback effect". In modified freezing, all relevant documents retrieved on the ith iteration and used for feedback on the $i + l$st iteration have their ranks frozen, and all nonrelevant documents ranked above the last ranked relevant document used for feedback are also frozen.

Table 2.2: Performance of the modified freezing of the results of different systems on the task of TRF. ▲ indicates that the improvements over no feedback and *all* the baseline feedback methods are statistically significant, at the 0.05 level using the paired two-tailed t-test with Bonferroni correction.

Method	**Robust04**			**WT10G**			**GOV2**		
	MAP	*P@20*	*RI*	*MAP*	*P@20*	*RI*	*MAP*	*P@20*	*RI*
MLE	0.2725	0.3949	n/a	0.2487	0.3136	n/a	0.3646	0.5318	n/a
SMM	0.3312	0.4829	0.55	0.2582	0.3812	0.31	0.4666	0.5760	0.51
DMM	0.3012	0.4638	0.42	0.2514	0.3609	0.19	0.4219	0.5621	0.42
RM3	0.3411	**0.5001**	0.63	0.3031	0.3849	0.36	0.4717	0.5851	0.55
RM4	0.3241	0.4766	0.37	0.2887	0.3705	0.31	0.4526	0.5781	0.45
RMM	0.3209	0.4719	0.56	0.2873	0.3760	0.34	0.4400	0.5639	0.57
MEDMM	0.3380	0.4891	0.53	0.3140	0.3920	0.34	0.4701	0.5891	0.61
SWLM	**0.3514**▲	0.4920	0.64	0.3155	**0.3976**	0.36	0.4813	**0.6016**▲	0.64
RSWLM	0.3434	0.4911	**0.68**	**0.3277**▲	0.3905	**0.39**	**0.4903**▲	0.5899	**0.69**

SWLM for True Relevance Feedback

True relevance feedback is employed to expand the user query based on either the explicit "relevant"/"non-relevant" judgments given by the user or implicit relevancy information inferred from the user behavior during his interaction with the system, for the top-k results returned by the retrieval system. In our experiments, we simulated this task. We consider the set of relevant documents on the top-10 results (first page of the search engine result page) in the ranked list as the documents judged as relevant by the user to form the feedback set. In our TRF experiments, like Lv and Zhai [185], we have removed queries that have no relevant documents in their top-10 results from the test collections. Information on the number of queries used for TRF in each collection is given in Table 2.1.

Table 2.2 presents the results of different systems on the TRF task. As can be seen, SWLM and RSWLM are best performing methods in terms of MAP and RI in all the collections and in terms of P@20 in the Web collections.

In the TRF task, although we use only documents that are explicitly labeled as relevant, since documents can be multi-topic, it is still possible that the feedback mechanism selects terms from non-relevant parts of the relevant documents. In the Robust04 collection, in which documents are not normally multi-topic, RM3 performs the best in terms of P@20. However, in the Web collections, which is more likely to contain multi-topic documents, SWLM, by controlling the effect of individual documents on the feedback model, significantly outperforms all the baselines.

Unlike the results in Table 2.3, in which RSWLM performs better than SWLM in terms of all metrics, in TRF, SWLM presents higher performance in terms

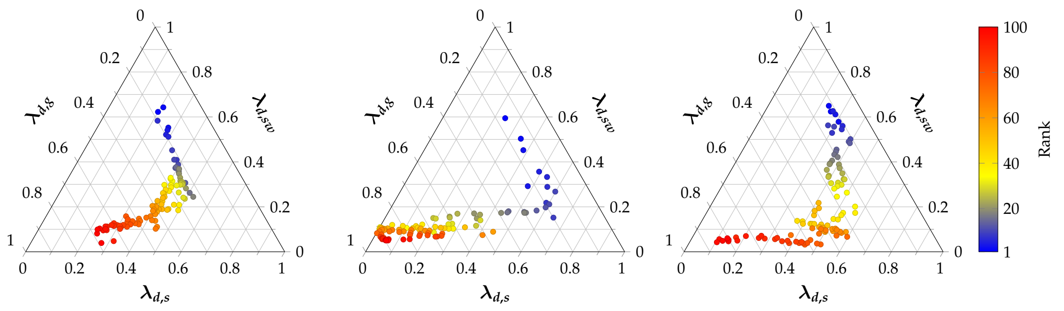
Table 2.3: Performance of different systems on the task of PRF. $^\blacktriangle$indicates that the improvements over no feedback (MLE) and *all* the baseline feedback methods are statistically significant, at the 0.05 level using the paired two-tailed t-test with Bonferroni correction.

Method	**Robust04**			**WT10G**			**GOV2**		
	MAP	*P@10*	*RI*	*MAP*	*P@10*	*RI*	*MAP*	*P@10*	*RI*
MLE	0.2501	0.4253	n/a	0.2058	0.3031	n/a	0.3037	0.5147	n/a
SMM	0.2787	0.4416	0.37	0.2193	0.3264	0.23	0.3214	0.5230	0.41
DMM	0.2701	0.4370	0.31	0.2184	0.3170	0.14	0.3026	0.5211	0.29
RM3	0.2937	0.4683	0.40	0.2406	0.3317	0.26	0.3417	0.5360	0.45
RM4	0.2690	0.4402	0.32	0.2323	0.3273	0.18	0.3316	0.5208	0.37
RMM	0.2681	0.4384	0.28	0.2222	0.3209	0.21	0.3112	0.5193	0.33
MEDMM	**0.2961**	**0.4719**	0.45	0.2413	0.3440	0.25	0.3396	0.5377	0.43
SWLM	0.2918	0.4674	**0.47**	0.2462	0.3377	0.28	0.3423	0.5316	0.50
RSWLM	0.2945	0.4704	**0.47**	**0.2506**$^\blacktriangle$	**0.3427**	**0.31**	**0.3510**$^\blacktriangle$	**0.5419**	**0.53**

of P@20. This might be due to the fact that in TRF, there is less noise and consequently less need to lead the feedback model toward the original query model. On the other hand, since RSWLM has no bias to the original query, it has the opportunity to retrieve some documents that are relevant without frequent occurrence of terms from the original query. Here, we presented the results of SWLM and RSWLM in the tasks of PRF and TRF compared to the baseline methods. We show that the new models are more effective than all previous methods, and also illustrated how they control the contribution of feedback documents in the feedback model based on their merit. Recall that SWLM takes a considerable risk by removing specific terms that are the most powerful retrieval cues if relevant, making the feedback task a critical experiment in distinguishing relevant and non-relevant terms. These results provide strong support for the effectiveness of significant words language models, and the general intuitions on the importance of building accurate models of relevance underlying them.

SWLM for Pseudo Relevance Feedback

Pseudo relevance feedback aims to expand the query to improve the performance of retrieval having no information about the judgments. In PRF, the underlying assumption is that the initially retrieved documents yield the relevant documents that can be used to refine the query. Thus, assuming the top-ranked documents $\mathcal{F} = \{d_1, \ldots, d_\mathcal{F}\}$ from the initial run as relevant, the feedback model $\theta_\mathcal{F}$ is estimated and used for the query expansion. Table 2.3 presents the results of employing significant words language models, regularized significant words language models as the feedback model as well as baseline methods on the task of PRF. As can be seen, RSWLM significantly



(a) Robust04 dataset (b) WT10G dataset (c) GOV2 dataset

Figure 2.5: Contribution of each of the relevance, general, and specific models in the top-100 documents as the feedback set, according to the λs learned in the RSWLM (the average over all the queries).

outperforms *all* the baselines in terms of MAP on WT10G and GOV2 collections, which are noisy Web collections.[7] Furthermore, it has the highest reliability of improvements in terms of the Robustness Index on all collections. On the PRF task, RSWLM works better than SWLM as it guides the estimator of the feedback model toward the query model and prevents it from being distracted by the noise from non-relevant documents.

Although it has been shown that PRF always improves the average performance of retrieval [125], under some parameter settings, for some topics it decreases the average precision. This is due to the fact that there might be some non-relevant documents in the feedback set containing non-relevant terms resulting in topic drift in the extracted feedback model [37, 129, 130]. Thus, as one of the main challenging problems in PRF, it is necessary to control the contribution of different feedback documents for inclusion in the feedback model based on their merit [130] for a specific query.

2.3.5 Relevance Decomposition

Significant words language models empower our proposed feedback method to dynamically determine the quality of each document. In Figure 2.5, as a sample, we take the top-100 documents as the feedback set and illustrate the average contribution of each of the significant words, general, and specific models in this documents, according to the λs learned in the regularized significant words language models.

It is an interesting observation that in all the collections the trend of the change in the contribution of three models is similar. In most cases, as the ranking goes down, the contribution of the significant words model decreases, which is in

[7]Note that we only indicate when (R)SWLM is significantly better than all baseline methods, they are always significantly better than the non-expansion MLE baseline.

accordance with the relevance probability of documents based on their ranking. However, this decay is slower for the Robust04 dataset than for WT10G and GOV2 datasets. This is likely because Robust04 dataset contains newswire articles, which are typically high-quality text data with little noise, in contrast to WT10G and GOV2, which are web collections containing a more heterogeneous set of documents.

Another interesting observation is that in all the collections we see that the top ranked documents are more likely to contain specific non-related terms than general non-related terms. In other words, as the rank of the document increases, the part of the document which is non-relevant becomes more general. One assumption would be that the retrieval models tend to rank documents with specific non-related terms higher than documents with general non-related terms. However, traditional retrieval models like KL-Divergence do not differ scores of documents based on their non-relevant part. Another hypothesis would be that the specificity or generality of non-related parts of documents is a matter of their length. For example, long documents are more probable to have specific non-related terms than short documents. We investigated the length of the retrieved documents based on their ranking in our experiments and, although the retrieval models in general might have some length bias [181], we observed no strong correlation between length and the ranking in our runs.

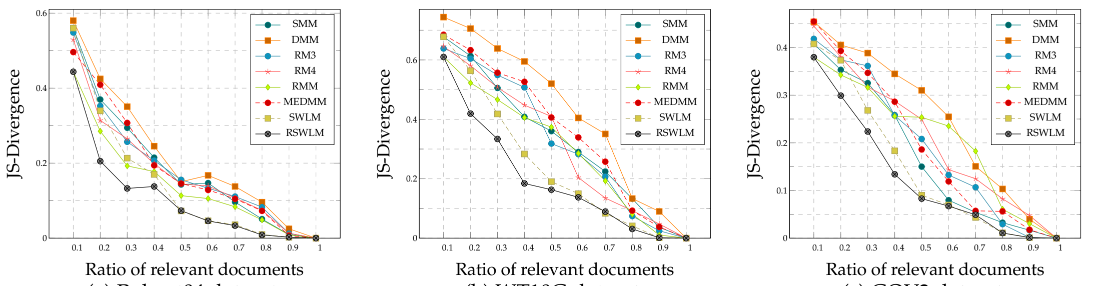
Based on the observation from Figure 2.5, we can conclude that the relevant component captured by the significant words dominates the ranking (as would be hoped and expected) and after that the specific component, and lastly the general component (in line with term weighting methods in the ranking models). These observations support that the proposed model is indeed more accurately modeling relevance than standard IR models. More generally, this analysis shows the analytic potential of the proposed model, for example to analyse the ranking of partially relevant or multi-topic documents, based on the generality or specificity of the subtopics involved, which we will defer to future work.

2.3.6 Robustness Against Noise

This section presents analyses resulting from experiments designed to study the robustness of our proposed feedback approach.

Divergence from Relevance

We designed an experiment to investigate the ability of the each feedback method to deal with noise in the PRF task, using top retrieved results. We measure the divergence of the estimated pseudo relevance feedback models,



(a) Robust04 dataset (b) WT10G dataset (c) GOV2 dataset

Figure 2.6: Divergence of true relevance feedback models and pseudo relevance feedback models in different systems, for queries with different ratio of relevant documents in top-10 results.

$\theta_{\mathcal{F}}^{prf}$, from the estimated true relevance models, $\theta_{\mathcal{F}}^{trf}$, that use only those documents from the top retrieved that are explicitly annotated as relevant. This experiment, in fact, study the extent to which a feedback method is able to learn a representation from a set of relevant and non-relevant documents which is similar to a representation learned using only the relevant documents.

To this end, we assume that $\theta_{\mathcal{F}}^{trf}$ is a model affected by the least amount of noise and calculate the JS-Divergence of $\theta_{\mathcal{F}}^{prf}$ and $\theta_{\mathcal{F}}^{trf}$ for all the approaches. To avoid the effect of the size of the models on the value of divergence, we take the top-500 terms of each model as the fixed length representation of the model.

Figure 2.6 shows the divergence of $\theta_{\mathcal{F}}^{prf}$ and $\theta_{\mathcal{F}}^{trf}$ for different groups of queries with different ratios of relevant documents among the top-10 documents, on all collections. As expected, for queries that only have a few relevant documents in the top-10, the divergence is high, and when all top-10 documents are relevant, two models perform similarly. For the Web collections, convergence of $\theta_{\mathcal{F}}^{prf}$ and $\theta_{\mathcal{F}}^{trf}$ is slower due to the fact that web documents are more noisy and it can be said that usually non-relevant retrieved documents are farther from relevant retrieved documents, compared to the Robust04 dataset. According to the plots in Figure 2.6, in all collections, SWLM and RSWLM have the least divergence in all the ratios. This means that our proposed models are more robust against being distracted by non-relevant documents. An interesting observation is that in all the collections, the behavior of SWLM and RSWLM is almost the same when at least half of the documents are relevant. In other words, we do not need regularization if at least half of the documents are of interest to the query's topic, either completely or partially.

Dealing with Poison Pills

Although it has been shown that on average, the performance of the results will be improved after applying feedback [125, 130], for some topics, employing

Table 2.4: Robustness of different systems against bad relevant documents based on $RI(D_r)$ measure.

Dataset	SMM	DMM	RM3	RM4	RMM	MEDMM	SWLM	RSWLM
Robust04	0.8663	0.7841	0.8716	0.8681	0.8843	0.8914	**0.9319**	**0.9305**
WT10G	0.8504	0.8190	0.8783	0.8961	0.8990	0.9082	**0.9583**	**0.9698**
GOV2	0.8456	0.8062	0.8809	0.8519	0.8910	0.8801	**0.9386**	**0.9209**

some documents may decrease the average precision of the initial run. As we discussed, in PRF, it could be due to the fact that the harmful feedback documents are not relevant. However, this can be the case for TRF, where a document that is labeled as relevant contains a set of off-topic terms and expanding the query with these terms when applying feedback leads to a decrease in the performance. The relevant documents that hurt the performance of retrieval after feedback are called "poison pills" [64, 125, 277, 299].

Terra and Warren [277] studied the effect of poison pills. They used a single relevant document for feedback with several systems to find documents that make the precision drop in all systems. They showed that more than 5% of all relevant documents perform poorly and in one third of all topics there exists at least one bad relevant document that can decrease the performance of the retrieval after applying feedback.

We have investigated this effect in the multiple feedback documents experiments. In these experiments, for each topic with more than ten relevant documents, we add relevant documents one by one, based on their ranking in the initial run, to the feedback set and keep track of the change in the performance of the feedback run after adding each relevant document to the feedback set compared to the feedback run without its presence in the feedback set.

To evaluate the robustness of different systems against bad relevant documents, we define a variant of *Robustness Index (RI)* [55] to be applicable at the document level instead of the topic level. For a set of relevant documents, D_r, the RI measure is defined as: $RI(D_r) = {N_r^+ - N_r^-}/{|D_r|}$ where N_r^+ and N_r^- denote the number of relevant documents that adding them to the feedback set, respectively helps or hurts the performance of the feedback. $|D_r|$ is total number of tested relevant documents. The higher the value of $RI(D_r)$ is, the more the method is robust against poison pills. Table 2.4 presents the $RI(D_r)$ of different systems on different datasets. As can be seen, both systems based on significant words language models are strongly robust against the effect of bad relevant documents in all datasets.

Furthermore, we have looked into the results of experiments on all the collections and extracted the set of poison pills, i.e., relevant documents whose adding to the feedback set decreases the performance of feedback in *all* the baseline systems. Overal, we found 118 poison pills and we observed that the

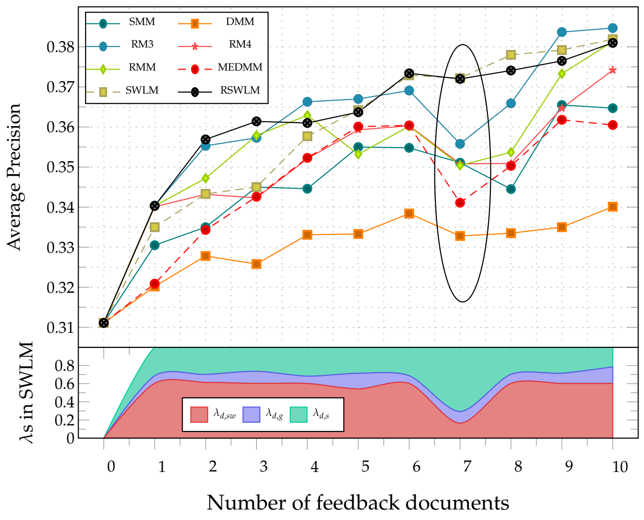


Figure 2.7: Dealing with poison pills: Effectiveness of different feedback systems faced with a poison pill (harmful relevant document) in topic 374 of TREC Robust04.

performance of RSWLM in these situations always has the smallest drop and in 92% of the cases, it provides the best average precision after adding the poison pill.

As discussed by Terra and Warren [277], poison pills are usually relevant documents that have either a broad topic, or several topics. In these situations, employing significant words language models enables the feedback system to control the contribution of these documents and prevents their specific or general terms from affecting the feedback model. Figure 2.7 shows how using the significant words language model empowers the feedback system to deal with the poison pills. In this figure, the performance of different systems on topic 374 in the Robust04 dataset is illustrated. As can be seen, adding the seventh relevant document to the feedback set leads to a substantial decrease in the performance of the feedback in all the systems. The query is “Nobel prize winners" and the seventh document is about one of the Nobel peace prize winners, Yasser Arafat, but at the end, it has a discussion concerning Middle East issues, which contains some highly frequent terms that are non-relevant to the query (see Figure 2.3). However, RSWLM and SWLM are able to distinguish this document as a poison pill and by reducing its contribution to the feedback model, i.e., learning a low value for $\lambda_{d_7,sw}$, they prevent the severe drop in the feedback performance.

So, our method inoculates the feedback model against poison pills by automatically determining whether adding a specific relevant document to the feedback set hurts the retrieval performance for a specific topic or not and controls its effect in the feedback model.

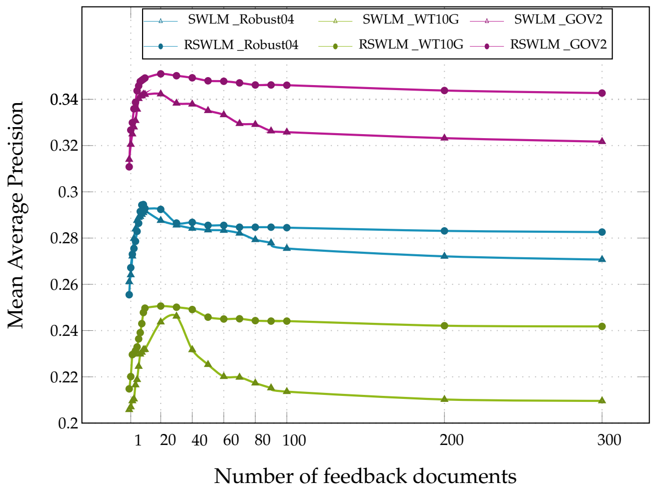


Figure 2.8: Sensitivity of SWLM and RSWLM to the number of feedback documents.

Sensitivity to the Number of Feedback Documents

In order to investigate the sensitivity of our proposed method to the number of documents in the task of PRF, which is a proxy to it sensitivity to the noisy documents in the feedback set, we set all other free parameters to the values that result in optimal average performance and plot the performance of SWLM and RSWLM with regard to the number of documents in the feedback set in Figure 2.8. Both methods have acceptable robustness. SWLM is more sensitive, especially on the Web collections, when low ranked documents are added, it is slightly affected by noises. However, RSWLM is strongly robust and less sensitive to the number of feedback documents.

Furthermore, according to Figure 2.8, the performance of both systems on all collections is the best when the number of feedback documents are around 10, which is a more or less the same observation in other feedback methods as well [186]. Moreover, this observation is in accordance with the information from the plots in Figure 2.5, in which the top-10 documents always possess a strong contribution of the significant words model, i.e., high values of $\lambda_{d,sw}$.

2.4 SWLM for Contextual Suggestion

Context is pervasive on the modern web, due to cloud-based and mobile applications, making every information access interaction part of an eternal user session. Effective ways to leverage this context are key to further enhancing the user experience, both in terms of better quality of results as in terms of easier ways to articulate complex information needs. This requires both effective ways of personalization to an individual user as well as customization to a profile based on groups of users.

For group level analysis, there is a need for extracting a group profile that captures the essence of the group, separate from the sum of the profiles of its individual members. This profile should be "specific" enough to distinguish the preferences of the group from other groups, and at the same time, "general" enough to capture all shared tastes, expectations, and similarities of its members.

Group profiling can help understand both explicit groups, like Facebook groups, and implicit groups, like groups extracted by community detection algorithms. One of the important applications of group profiling is in the contextual suggestion problem [126]. Contextual suggestion is the task of searching for complex information needs that are highly dependent on both context and user interests. This task is defined in the form of the personalized point of interest recommendation task, in which the recommender system provides a ranked list of suggestions given the profile of the user and the context in which the user seeks for the suggestion.

Using individual preferences for contextual suggestion is not always possible. For example, sometimes there is a new user in the system with no historical interactions and no rich information about the preferences,[8] or sometimes the user is not able to determine his/her preferences explicitly. In these situations, group based contextual suggestion would be beneficial to augment the user's profile and suggest content to the user based on the preferences of the groups that the user belongs to.

In contextual suggestion, given the information of users including their age, gender, and set of rated places or activities as the user preferences (ratings are in the range of -1 to 4), the task is to generate a list of ranked suggestions from a set of candidate attractions, by giving the user information as well as some information about the context, including location of trip, trip season, trip type, trip duration, and the type of group the person is travelling with.

We employed SWLM for group profiling to be able to employ group information in the contextual suggestion task [67, 69, 128]. We use group profiles estimated by SWLM with respect to the different grouping criteria and investigate how group-based information helps to improve the general performance on contextual suggestion task.

In the rest of this chapter, we address our third and last research question of this chapter:

> **RQ-1.1.3** *How well can significant words language models profile groups of entities and how effective are these profiles in content customization tasks?*

[8] Cold start problem in contextual suggestion.

Table 2.5: Statistics of users groups resulted by grouping based on different critera.

Grouping Criterion	**Age**					**Gender**		**Group Type**				**Trip Type**			**Trip Duration**				**Season**			
Groups	< 20	$20-30$	$30-40$	$40-50$	> 50	male	female	alone	family	friends	other	holiday	business	other	night out	day trip	weekend	longer	spring	summer	autumn	winter
Group Size (#users)	9	87	60	22	23	107	91	23	101	65	4	176	7	12	8	17	99	74	20	99	52	22

We explain how to use SWLM as a profiling approach to represent a group of entities and use this representation to improve the quality of contextual suggestion task.

2.4.1 Experimental Setup

We use of the TREC 2015 Contextual Suggestion[9] batch task dataset to evaluate the effectiveness of SWLM as a profiling approach for the contextual suggestion task. The dataset contains information from 207 users including their age, gender, and set of rated places or activities as the user preferences (rates are in the range of -1 to 4). The task is to generate a list of ranked suggestions from a set of candidate attractions, by giving the user information as well as some information about the context, including location of trip, trip season, trip type, tripe duration, and the type of group the person is travelling with. Based on the information in the dataset, we divide users into several groups. Groupings are based on the users information and context information. Table 2.5 presents grouping criteria, the groups, and number of users in each group.

2.4.2 SWLM for Group Profiling

We generate group-based rankings of suggestions based on the group profiles estimated by SWLM to evaluate the quality of the estimated group profiles in contextual suggestion task. To this end, first we choose one of the grouping criteria mentioned in Table 2.5 like users' age. Then we estimate a language model representing each group as its profile using SWLM. Afterward, regarding the information of the given request, i.e., the user information and context information, the group which the user belongs to is selected and based on the profile of this group as well as the language model of the candidate, we rank the suggestions.

Beside the group-based ranking, we generate a ranked list of suggestions based on the preferences of the users, regardless of their group memberships, as a baseline. To do so, a language model is estimated using the mixture of the language model representing user preferences weighted by their ratings. Then,

[9] *https://sites.google.com/site/treccontext/trec-2015*

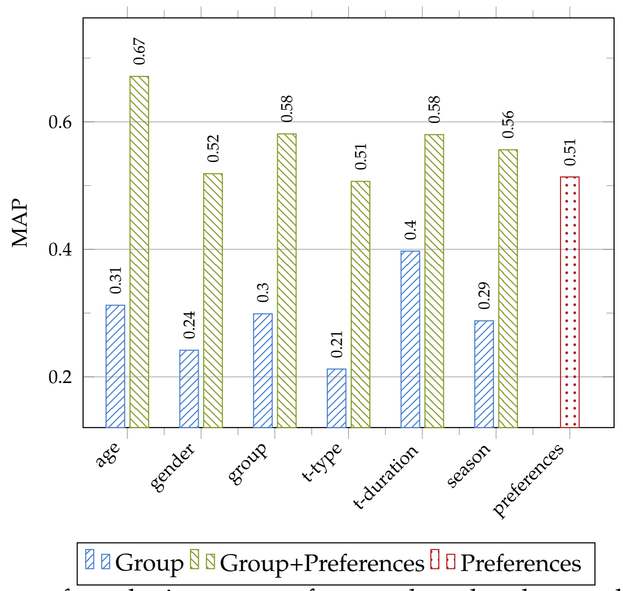


Figure 2.9: Performance of employing user preferences-based and group-based customization on contextual suggestion task. Improvements of combining group-based and preferences-based approach over the preferences-based approach and corresponding group-based approaches are statistically significant based on one-tailed t-test, with p-value < 0.05.

based on the similarity of the preferences language model and the candidate language model, we rank the suggestions.

Using SWLM, we learn the contribution of each of *specific*, and *general*, *group* (which is in fact *significant words*) models, i.e., $\lambda_{user,s}$, $\lambda_{user,g}$, and $\lambda_{user,sw}$. Having these parameters enables us to efficiently combine the group-based model with the preferences-based model for content customization. To this end, we smooth the preferences-based model of the user with both the group model and the general model using JM-smoothing [330] employing the learned parameters.

Figure 2.9 presents the performance of: 1) a system that provides suggestions based on users preferences, 2) different systems that take different grouping criteria and provide group-based suggestions, 3) different systems that take different grouping criteria and combine group-based suggestions with user-based suggestions.

Among the group-based strategies, suggestions based on the duration of the trip is the most effective strategy. Also age of the user and the type of the group the user travels with, are rather important while type of the trip is not so important. This could be due to the fact that most of the time, the user's interests and attractions do not change based on the type of trip which could be "business" or "holiday." On the other hand, combining the preferences-based suggestions with group-based suggestions in all grouping strategies leads to improvement. This means that in case of incompleteness of a user's profile, customizing the content based on the groups that user belongs to, implicitly fills the missing

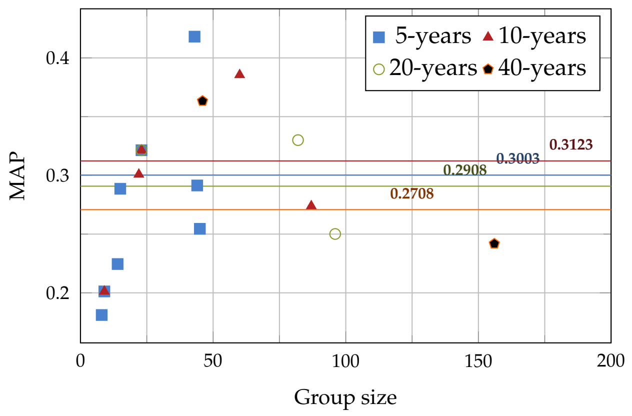


Figure 2.10: Effect of groups granularity on the performance group profiling.

information and improves the performance of the suggestions. However, this depends on the quality of the group profiles that should reflect significant (not general, not specific) characteristics of the groups.

2.4.3 Effect of Group Granularity

In the grouping stage, sometimes users can be grouped based on different levels of granularity. For example, having the age of users, discretization can be done using binning with different sizes of bin. In this section, we analyse the effect of granularity of groups, and consecutively the size of the groups with a fixed volume of train data, on the quality of group profiling.

We have selected "age" of users as the grouping criterion and tried different bin sizes for discretization: 5 years, 10 years, 20 years and 40 years. Figure 2.10 shows the quality of groups profiles on different levels of granularity and consequently on different sizes of groups in the task of contextual suggestion. Each point in the figure represents a group of users and its position determines its size and the performance of group-based contextual suggestion for the users within the group. Moreover the horizontal lines represent overall performance of different levels of granularity. As can be seen, since the number of sample users is limited fine-grained grouping leads to having smaller groups. So small number of samples affects the group profiling quality and slightly decreases the performance. While coarse-grained grouping leads to having large groups that leads to not being able to adequately customize the group profile.

In our dataset, 10 years granularity for "age" has the best performance since the formed groups are big enough so that the group profiling approach is able to estimate high quality models, and they are small enough so that the group profiles are easily distinguishable which leads to a more effective customization.

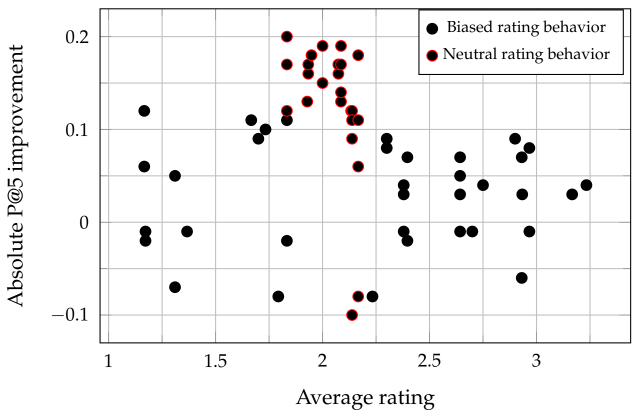


Figure 2.11: Improvement of the performance of contextual suggestion using group profiles for users with different rating behavior.

2.4.4 Effect of Rating Behavior

As we showed, using group-based information that is modeled by SWLM helps to improve the performance of contextual suggestion. To study where this improvement comes from, we looked into the data to see in which cases adding group information helps and in which cases it is not effective. We observed that there is a correlation between the amount of improvement in contextual suggestion using group-based information and the rating behavior of users.

Figure 2.11 shows the scatter plot of the change in $p@10$ after employing group-based information based on different rating tendency. According to the plot, group-based information works better when the users have a neutral tendency in their rating (around rate 2) and it is less likely to help when users have rather strong biases by rating attraction with high or low rates. This could be due to the fact that in case of having neutral users, we have less strong information coming from their profile and then group-based information is compensating this lack of strong signals.

2.5 Related Work

In this section, we discuss related studies to the applications that we evaluated SWLM on, i.e, in feedback in the retrieval task and group profiling in contextual suggestion.

2.5.1 Relevance Feedback

It has been shown that there is a limitation on providing increasingly better results for retrieval systems only based on the original query [286]. So, it is crucial to reformulate the search request using terms that reflect the user's information need to improve the performance of the retrieval systems. To address this issue, automatic feedback methods for information retrieval were introduced fifty years ago [234] and have been extensively studied during past decades. As the earliest relevance feedback approach in information retrieval, the Rocchio method [234] has been proposed in the vector processing environments for changing the query vector to be similar to the relevant documents vectors and dissimilar to the non-relevant documents vectors. Later, probabilistic methods have been proposed to select expansion terms from feedback documents based on a term weighting approach [233, 285]. With the development of language models, several feedback approaches have been proposed in this framework to improve the query language model [136, 169, 187, 274, 329]. The mixture model [329] is one of the well-known feedback methods in the language modeling framework, which empirically performs well. The idea is to extract a discriminative language model of feedback documents by decreasing weights of the background terms. As an extension to this model, the regularized mixture model has been proposed by Tao and Zhai [274], which not only involves the query model in the estimated feedback model but also has document-specific mixing coefficients to let different documents have a different amount of background terms.

In the relevance model (RM) [2, 169] given the query, a model is estimated as a multinomial distribution over terms that indicates the likelihood of each term given the query as the evidence, based on the occurrences of the term together with the query terms in the feedback documents. In a comparable study conducted by Zhai and Lafferty [329], it has been shown that RM3 as a variant of the relevance model is one of the best performing methods which is strongly robust. Divergence minimization [329] tries to estimate a feedback model that is close to the language model of every feedback document but far from the collection language model as an approximation of the non-relevant language model. This method generates a highly skewed feedback model, which makes it unable to perform well. Lv and Zhai [187] proposed the maximum-entropy divergence minimization model that, by adding an entropy term, regularizes the original divergence minimization model leading to significant improvements in the performance of the original method.

The parsimonious language model [136] is one of the models employed for feedback [135, 157, 194]. It tries to describe the feedback model using a smaller number of parameters. Similar to the mixture model, the common words in the collection are removed from the model in the estimation process, which leads to a more lean and mean language model. Zamani and Croft [320], considering the feedback problem as a recommendation problem, made use of matrix factorization in order for predicting expansion terms in a weighted manner.

Besides the ad hoc studies, there have been some initiatives aimed at studying the problem of (pseudo-)relevance feedback in detail. In 2003, Reliable Information Access (RIA) Workshop [125, 299] was organized with the goal of understanding the contributions of both system variability factors and topic variability factors to the overall retrieval variability in feedback. In addition to the Relevance Feedback track, the Robust track in TREC defined one of the goals to improve the consistency of feedback systems by focusing on the poorly performing topics [295].

Arguably, the key issue in feedback is robustness in terms of being able to deal with non-relevant terms from non-relevant or partially relevant documents. SWLM addresses the robustness problem head on. This is achieved by using the information from the collection and other feedback documents to control the contribution of documents in the feedback model regarding their merit and to avoid the selection of non-relevant expansion terms.

2.5.2 Group profiling for Content Customization

Group profiling can help understand both explicit groups, like Facebook groups, and implicit groups, like groups extracted by community detection algorithms. There is a wide range of applications for group profiling, like understanding social structures [272], network visualization, recommender systems [5, 143, 252], and direct marketing [60].

There is various research done on the task of group profiling which, given the individual attributes and preferences, aims to find out group-level shared preferences [192, 247]. Tang et al. [272] presented three methods for group profiling: *Aggregation*, which tries to find features that are shared by the whole group; *Differentiation*, which tries to extract features that can help to differentiate one group from others; and *Egocentric differentiation*, which tries to extract features that can help to differentiate members of one group from the neighbour members. In recent work, Hu et al. [143] proposed a deep-architecture model to learn a high level representation of group preferences.

For group recommendation, there is research on building a model of a group by forming a linear combination of the individual models [148]. Some of them construct the group's preference model on the basis of individual preference models, using a notion of distance between preference models [318]. Some approaches try to divide the group into several categories of homogeneous users and specify the preference model for each subgroup. Then they create the group model as a weighted average of the subgroup models, with the weights reflecting the importance of the subgroups [9].

2.6 Conclusion

In this chapter, we focused on addressing **RQ-1.1**: *"How to learn robust representations for entities and abstract concepts that are affected by neither undiscerning general, nor noisy accidental features, given the structural relations in the data?"*. Inspired by a discussion on the early work by Luhn [183] about *significant words*, we address **RQ-1.1.1** by proposing *significant words language models* (SWLM) to estimate a representation for a set of documents that captures significant terms by avoiding the distracting effect of common observation as well as rare observation.

To investigate **RQ-1.1.2**, We utilized SWLM for the relevance feedback problem and showed that as the feedback model, it presents promising performance on both true and pseudo relevance feedback. Analyzing the results, we indicated that the strength of SWLM in feedback is due the fact that it is capable of controlling the contribution of feedback documents in the feedback model based on their level of relevancy, which copes with the problem of topic drift in query expansion. We assessed the robustness of SWLM in different experiments and showed that in PRF, SWLM has the least vulnerability against noise in the data both at term-level and document-level. We also employed SWLM as a group profiling approach for the task of contextual suggestion to study **RQ-1.1.3**. Our experimental results showed that using the group representations estimated by SWLM, we can improve the performance of content customization.

We named our model, significant "words" language model in honor of Luhn, however, it could be employed in non-textual environments, since in general, the idea is to extract significant "features" representing the shared essence of a group of entities.

The process of estimating SWLM leads to a sparse model, i.e., θ_{sw} assigns zero probability to many terms that are identified as either too general or too specific for representing an entity. Sparsity is a desirable property, for instance, sparse representations are easier to interpret or more likely to posses separability which is important for collision resistance hash functions. In Chapter 3, we extend SWLM to hierarchically structured entities and discuss separability as a key objective for learning the representations.

3 Representational Separability for Hierarchically Structured Data

Hierarchies are powerful structures to model different levels of associations in the data. They can evolve over time in terms of the relations between entities in the hierarchy. Learning representations that are agnostic to these changes is not trivial. We can, however, take the horizontal and vertical dependencies in a hierarchy into account and learn highly separable representations for entities, that are less sensitive to the structural changes and more transferable over time.

3.1 Introduction

Hierarchies are effective and common structures for representing information, and many domains are naturally organized in a hierarchy. Organizing data in a hierarchical structure is valuable since it determines relationships in the data at different levels of resolution and picks out different categories relevant to each of the various layers of memberships. Besides that, using hierarchies is an effective way of representing the information as it eases the task of *comparison*, which is a critical factor in analogical reasoning [142]. For example, the problem

This chapter is based on [68, 70].

of deciding whether two entities are analogous can be formalized as the problem of checking the level of abstraction at which these entities are instances of the same node in a hierarchy [3].

Taking advantage of the structure in a hierarchy requires modeling and representing entities, taking their relationship in the hierarchy into consideration. There are two types of dependencies in the hierarchies: i) *Horizontal dependency*, which refers to the relations of entities in the same layer. A simple example would be the dependency between siblings which have some commonalities in terms of being descendants of the same entity. ii) *Vertical dependency*, which addresses the relations between ancestors and descendants in the hierarchy. For example, the relation between: root and other entities.

Due to the existence of two-dimensional dependencies between entities in the hierarchy, modeling them regardless of their relationships might result in overlapping representations that are not capable of making different entities distinguishable. Learning representations with minimal overlap is one of the requirements for collision resistance systems and when the representations are not well-separated, classification and retrieval systems are less likely to work well [174]. Thus, *two-dimensional separability*, i.e., *horizontal and vertical separability*, is one of the key requirements of hierarchical classification.

In this chapter, we focus on one of our research questions:

> **RQ-1.2** *How to learn separable representations for hierarchically structured entities that are less sensitive to structural changes in the data and more transferable across time?*

We introduce hierarchical significant words language models, which extend the idea of significant words language models to hierarchical structures. We assume that entities, like people, organizations, concepts, and ideologies are organized in a hierarchy, and for each entity, there is textual data associated with the entity with respect to the subhierarchy under the entity, where the data is generated only at the leaves of the hierarchy. Each entity, as a node in the hierarchy, is represented by a specific probabilistic language model.

Given the hierarchical structure and the data associated with entities in the hierarchy, HSWLM iteratively sparsifies the representation of the entities by discarding features that are well explained by their ancestors, i.e., general features, as well as features that reflect the characteristics of individual descendants, i.e., specific features. This leads to representations for entities that are both vertically and horizontally separable, in terms of their position in the hierarchy, as they capture only the significant features of entities.

As a concrete example, consider a simple hierarchy of a multi-party parliament as shown in Figure 3.1, which determines different categories relevant to the

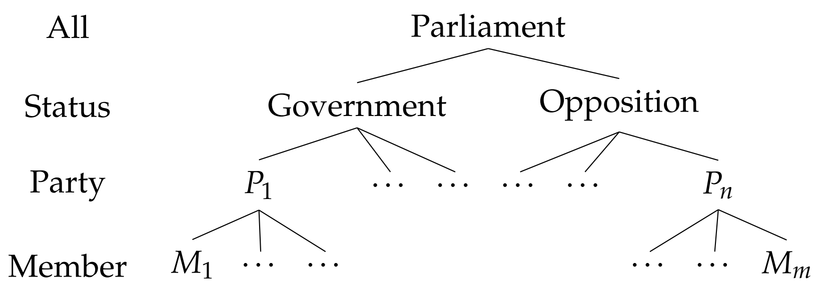
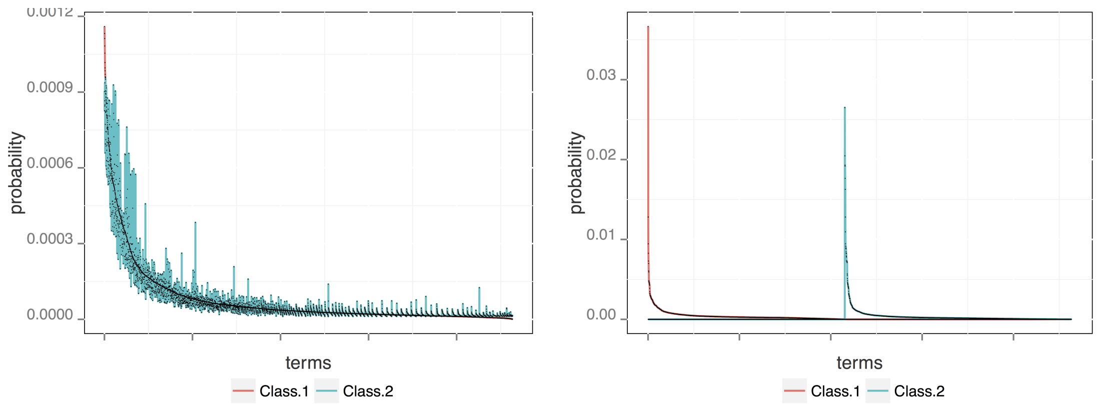


Figure 3.1: Hierarchical relations in parliament.

different layers of membership in the parliament. We can associate an individual member of parliament by her speeches, a political party by their member's speeches, the opposition by the speeches of members of opposition parties, etc. In order to represent a party in this hierarchy, a proper representation would show common characteristics of its members —not members of other parties (*horizontal* separation), and capture the party's generic characteristics— not unique aspects of the current members captured in the individual member's layer or aspects of whether the party is in government or opposition captured in the status layer (*vertical* separation).

3.1.1 The Importance of Representational Separability

The concept of *separability* is of crucial importance especially when the task is not just ranking a set of items, but making a boolean decision about the labels of each item in the set. Regarding this concern, Lewis [175] has presented the Probability Threshold Principle (PTP), as a stronger version of the Probability Ranking Principle [231], for binary classification, which discusses optimizing a threshold for separating items regarding their probability of class membership. PTP is a principle based on the separability in the score space. However, here we discuss separability in the data representation and later in this chapter we define a *Strong Separation Principle* as the counterpart of PTP in the feature space. Separation in the data or feature space is a favorable property that not only helps to improve for ranking or classification algorithms but also brings out characteristic features for human inspection. Figures 3.2a and 3.2b illustrate two different ways of representing two entities in the status layer of the parliamentary hierarchy, i.e., government and opposition. Each representation is a probability distribution over terms (a language model) based on the speeches given by all the members in the corresponding status. In each figure, we sort the terms based on their weights in one of the representations and plot the other in the same order. As can be seen, although distributions over terms in Figure 3.2a for two classes are different, they do not suggest highly separable



(a) A non-separable representation of data. (b) A well-separable representation of data

Figure 3.2: Probability distribution over terms for data in two different classes, (entities in the statues layer of the parliament), sorted based on the term weights in one of the classes.

representations for classes. However, estimated language models in Figure 3.2b provide highly separable distributions over terms for two classes, identifying the characteristic terms that uniquely represent each class, and can be directly interpreted.

Besides effectiveness and intractability, two-dimensional separability in representations of hierarchical entities increases the robustness of these representations against changes in the structure of the hierarchy. In other words, when learning a representation for an entity in a hierarchy, if we remove features that are based on the dependencies between this entity and other entities, we will capture only the solid set of features reflecting the main characteristic of this entity. This means that changes in the structure, i.e., in relations between entities, will not affect their learned representations.

As an example of the importance of robust representations in evolving hierarchies, assume we would learn a representation for the "US president" over the current data. It is obvious that we need to distinguish the role in office from the person who is the current president; otherwise the learned representation would not be valid after the next election. If we can separate the representation of the function from the representation of the person fulfilling it, for example by abstracting over several presidents, that would in principle be robust over time.

3.1.2 Detailed Research Questions

We break down our main research question in this chapter into three concrete research questions:

> **RQ-1.2.1** *What makes separability of representations a desirable property for classifiers?*

> **RQ-1.2.2** *How can we estimate horizontally and vertically separable representations for hierarchically structured entities?*
>
> **RQ-1.2.3** *How can separability of representations for hierarchical entities improve their transferability?*

In the following sections, we will address these research questions.

3.2 Separability in the Hierarchies

In this section, we explore the first research question in this chapter:

> **RQ-1.2.1** *What makes separability of representations a desirable property for classifiers?*

In addition to the investigation of the separation property as a general foundational property of classification and defining a *Strong Separation Principle*, we discuss the two-dimensional separation property of hierarchical classification.

3.2.1 Separation Property

Separability is a highly desirable property for constructing and operating autonomous information systems [175], and especially classifiers. Here, we present a step by step argument which shows that based on the classification principles, having better separability in the feature space leads to better accuracy in the classification results.

Based on the *Probability Ranking Principle (PRP)* presented by Robertson [231], Lewis [175] has formulated a variant of PRP for binary classification:

> *For a given set of items presented to a binary classification system, there exists a classification of the items such that the probability of class membership for all items assigned to the class is greater than or equal to the probability of class membership for all items not assigned to the class, and the classification has optimal expected effectiveness.*

Since in many applications, autonomous systems need to decide how to classify an individual item in the absence of entire items set, Lewis has extended the PRP to the *Probability Threshold Principle (PTP)*:

> *For a given effectiveness measure, there exists a threshold p, $0 < p < 1$, such that for any set of items, if all and only those items with probability of class membership greater than p are assigned to the class, the expected effectiveness of the classification will be the best possible for that set of items.*

PTP, in fact, discusses optimizing the effectiveness of classifiers by making items separable regarding their probability of class membership, which is a discussion on "separability in the score space." Based on PTP, optimizing a threshold for separating items is a theoretically trivial task; however, there are practical difficulties.

The main difficulty refers to the fact that retrieval models are not necessarily capable of measuring actual probabilities of relevance for documents [8], so they do not guarantee to generate a set of scores from which the optimum cutoff can be inferred. In this regard, a great deal of work has been done on analyzing the score distribution over relevant and non-relevant documents to utilize this information for finding the appropriate threshold between relevant and non-relevant documents [7, 8, 156]. The more the score distributions of relevant and non-relevant documents are separable, the easier it is to determine the optimum threshold. So, obtaining the *separation property* in the score distributions of relevant and non-relevant documents is one of the key focus areas for retrieval models.

There are two ways to obtain separability in the score distributions. We could address the complex underlying process of score generation and investigate ranking functions that yield a separable score distribution, as in the score distributional approaches [8]. Alternatively, we can investigate ways to provide existing scoring functions with a highly separable representation of the data. That is, the "term distribution" directly provides information about the "probability of relevance" [58] and if there are separable distributions over *terms* of relevant and non-relevant documents, a scoring function satisfying PRP will generate scores that separate the classes of relevant and non-relevant documents. Thus, a *separation property* on feature distribution for representing the data is a favorable property, which follows a better accuracy of classifiers' decisions.

In this chapter, we investigate the role of separation in the term or feature spaces, in which we introduce a formal definition for separability and formulate a principle on the effectiveness of classification based on separation property and leave a more formal treatment to future work.

As a formal and general definition, we can refer to representation separability as follows:

DEFINITION 3.1. *The representation of an entity is "separable" if, and only if, it has unique, non-overlapping features that distinguish it from other representations.*

We argued that separability in feature space leads to separability in score space. Based on this and the given definition of separability, we present the *Strong Separation Principle (SSP)*, which is a counterpart of the PTP [175] in the feature space:

> *For a given set of items presented to a classification system, for each class there exists at least one feature value δ in the representation of items, and a threshold τ, such that for any set of items, if all and only those items with $\delta > \tau$ are assigned to the class, the classification will have the optimal possible performance for that set of items in terms of a given effectiveness measure.*

SSP, in general, is a stronger version of PTP. In strict binary classification, if you have PTP, which holds on the whole feature space, SSP will be satisfied, however in the multi-class case, SSP is stronger and it implies PTP, but not the other way around.

Based on PTP, there is always an underlying probabilistic scoring function on the set of all features, which generates membership probabilities as the scores of items. These scores make items separable with regards to a threshold. So, the scoring function can be deemed as a mapping function which maps items to a new feature space in which the score of each item is a single feature representation of that item (membership probabilities, i.e., scores, in PTP would be equivalent to δ in SSP). Thus, when the SSP holds, the PTP and PRP will also hold.

One could consider a stronger version of the SSP in which "all" the features in the representations need to be non-overlapping, but the SSP is sufficient for optimizing the effectiveness of the classifier. The separation principle can be formally extended to hierarchical classification in a straightforward way. In the rest of this section, we will discuss the separation property for the hierarchical classification and explain how to estimate separable representations with the aim of satisfying SSP in order to improve the classification effectiveness.

3.2.2 Horizontal and Vertical Separability

In hierarchically structured data, there are two main types of boundaries existing in the data, horizontal boundaries, and vertical boundaries [68]. Hence, a separation property should be established in two dimensions. This means that not only separation between entities' representation in one layer is required, but a concept related to separation between the distribution of terms in different layers is needed.

The separation between entities in the same layer is a related concept to the fundamental goal of all classifiers on the data with a flat structure, which is making the data in different classes distinguishable [246]. However, separation between entities in different layers is a concept related to difference of abstraction level and modeling data in different layers in a separable way can help the scoring procedures to figure out the meaning behind the layers and make their decisions less affected by the concepts of other unrelated layers, thus leading to conceptually cleaner and theoretically more accurate models.

Based on Definition 3.1, we formally define horizontal and vertical separability in the representation of hierarchically structured entities as follows:

DEFINITION 3.2. *The representation of an entity in the hierarchy is "horizontally separable" if, and only if, it is* separable *compared to other entities in the same layer, with the same abstraction level.*

DEFINITION 3.3. *The representation of an entity in the hierarchy is "vertically separable" if, and only if, it is* separable *compared to other entities in the other layers, with different abstraction levels.*

To formalize these concepts, consider an example where we have a simple three layer hierarchy of text documents with "IsA" relations, where the individual documents take place in the lowest layer, and each node in the middle layer determines a category, representing a group of documents, i.e., its children, and the supernode at the top of the hierarchy deemed to represent all the documents in all the groups in the hierarchy. There is a key point in this hierarchy to which we will refer for learning representations for the hierarchical entities: "each node in the hierarchy is a general representation of its descendants."

First assume that the goal is to estimate a language model representing category c, as one of the entities in the middle layer of the hierarchy, and we need the learned representation to possess *horizontal separability*. To estimate a horizontally separable representation of a category, which represents the category in such a way that it is distinguishable from other categories in the middle layer, the key strategy is to eliminate terms that are common across all the categories (overlapping features) and preserve only the discriminating ones.

To do so, we consider there is a general model that represents all the *common* terms of all the categories in the middle layer, θ_c^g. Also, the standard language model of c is the model estimated from concatenation of all the documents in c using MLE, θ_c. We assume that θ_c is drawn from the mixture of the *latent horizontally separable model*, θ_c^{hs}, and the general model that represents shared terms of all categories, θ_c^g:

$$p(t|\theta_c) = \lambda p(t|\theta_c^{hs}) + (1-\lambda) p(t|\theta_c^g), \tag{3.1}$$

where λ is the mixture coefficient. Regarding the meaning of the relations between nodes in the hierarchy, the top node in the hierarchy is supposed to be a general representation of all categories. On the other hand, θ_c^g, is supposed to represent the general features of all the categories in the middle layer. Thus, we can approximate θ_c^g with the estimated model of the top node in the hierarchy, θ_{all}:

$$p(t|\theta_c) \approx \lambda p(t|\theta_c^{hs}) + (1-\lambda)p(t|\theta_{all}). \tag{3.2}$$

We estimate θ_{all} using MLE as follows:

$$p(t|\theta_{all}) = \frac{tf(t,all)}{\sum_{t'} tf(t',all)} = \frac{\sum_{c\in all}\sum_{d\in c} tf(t,d)}{\sum_{c\in all}\sum_{d\in c}\sum_{t'\in d} tf(t',d)}, \tag{3.3}$$

where $tf(t,d)$ indicates the frequency of term t in document d and θ_{all} is in fact the collection language model.

Now, the goal is to extract θ_c^{hs}. With regard to the generative models, when a term t is generated using the mixture model in Equation 3.2, first a model is chosen based on λ and then the term is sampled using the chosen model. The log-likelihood function for generating the whole category c is:

$$\log p(t|\theta_c^{hs}) = \sum_{t\in c} tf(t,c)\log\left(\lambda p(t|\theta_c^{hs}) + (1-\lambda)p(t|\theta_{all})\right), \tag{3.4}$$

where $tf(t,c)$ is the frequency of occurrence of term t in category c. With the goal of maximizing this likelihood function, the maximum likelihood estimation of $p(c|\theta_c^{hs})$ can be computed using the Expectation-Maximization (EM) algorithm by iterating over the following steps:
E-step:

$$e_t = tf(t|c)\cdot\frac{\lambda p(t|\theta_c^{hs})}{\lambda p(x|\theta_c^{hs}) + (1-\lambda)p(x|\theta_{all})}, \tag{3.5}$$

M-step:

$$p(x|\theta_c^{hs}) = \frac{e_t}{\sum_{t'\in\mathcal{V}} e'_t}, \text{ i.e., normalizing the model,} \tag{3.6}$$

where $\mathcal{V}$ is the set of all terms with non-zero probability in θ_c. In Equation 3.5, θ_c is the maximum likelihood estimation of category c: $p(t|\theta_c) = {\sum_{d\in c} c(t,d)}/{\sum_{d\in c}\sum_{t'\in d} c(t',d)}$ and θ_c^{hs} represents the horizontally separable model, which is initialized by the maximum likelihood estimation in the first iteration, similar to θ_c.

Considering the above process, a horizontally separable model is a model that is **specified** by taking out general features that have a high probability in "all" categories, i.e., collection language model, which is similar to the concept of the parsimonious language model, introduced by Hiemstra et al. [136].

Now assume that we want to extract a language model possessing *vertical separability* for the category c, i.e., a representation that makes this category distinguishable from entities both in the lower layer (each individual document) and the top layer (collection of all documents). In the procedure of making the representation horizontally separable, we argued that we can reduce the problem to removing terms representing the top node, which results in a representation that is separable from the top node in the upper layer. This means that we are already half-way towards making a representation vertically separable; thus, the representation only requires it to be made separable from its descendant entities in the lower layer. Recall that the representation of each node is, in fact, a general representation of all its descendants. So making the representation of a category separable from its descendant documents can be translated to removing terms that describe individual documents, but not reflect the shared commonalities of all descendant documents in that category. We call these terms, document specific terms.

For each category c, we assume there is a model, θ_d^s, that captures document specific terms, i.e., terms from documents in that category that are good indicators for individual documents but not supported by all of them. Also, we assume that the standard language model of c, θ_c, is drawn from the mixture of the *latent vertically separable model*, θ_c^{vs}, and θ_d^s:

$$p(t|\theta_c) \;=\; \lambda p(t|\theta_c^{vs}) + (1-\lambda)p(t|\theta_d^s), \tag{3.7}$$

where λ is the mixing coefficient. We estimate θ_d^s using the following equation:

$$p(t|\theta_d^s) \xleftarrow{normalized} \sum_{d_i \in c} \left(p(t|\theta_{d_i}) \prod_{\substack{d_j \in c \\ j \neq i}} (1 - p(t|\theta_{d_j})) \right), \tag{3.8}$$

where $p(t|\theta_{d_i}) = {c(t,d_i)}/{\sum_{t' \in d_i} c(t',d_i)}$. This equation assigns a high probability to a term if it has high probability in one of the document models, but not others, marginalizing over all the document models. This way, the higher the probability is, the more specific the term will be. Now, the goal is to extract the θ_c^{vs}. An EM algorithm, similar to Equations 3.5 and 3.6 can be applied for estimating θ_c^{vs} by removing the effect of θ_d^s from θ_c.

Considering the above process, a vertically separable representation is a representation that is **generalized** by downweighting specific terms that have a high probability in one of the descendant documents, but not others.

3.2.3 Two-Dimensional Separability

In order to have fully separable representations in hierarchical classification, they should own two-dimensional separation property. We define two-dimensional separability as follows:

DEFINITION 3.4. *The representation of an entity in the hierarchy is "two-dimensionally separable" if, and only if, it is both horizontally and vertically separable at the same time.*

Intuitively, if a representation of an entity is two-dimensionally separable, it should capture *all*, and *only*, the essential features of the entity taking its relative position in the hierarchy into consideration. In the next section, we will discuss how to estimate two-dimensional separable representations for entities in hierarchies with more than three layers.

3.3 Hierarchical Significant Words Language Models

In this section, we address the second research question of this chapter:

> **RQ-1.2.2** *How can we estimate horizontally and vertically separable representations for hierarchically structured entities?*

We introduce Hierarchical Significant Words Language Models (HSWLM), which is an extension of Significant Words Language Model that is explained in Chapter 2 tailored to textual hierarchical entities. HSWLM is, in fact, a particular arrangement of multiple passes of the procedures of making representations of hierarchical entities vertically and horizontally separable, as they are explained in Section 3.2.2. We use the idea of parsimonious language models [136] to parsimonize the representation of an entity by down-weighting terms that do not reflect the significant features of that entity, with regards to its position in the hierarchy. In the parsimonious language model, given a raw probabilistic estimation, the goal is to reestimate the distribution so that non-essential parameters of the raw estimation are down-weighted if they are well explained in a given background distribution. The proposed approach for estimating hierarchical significant words language model iteratively reestimates the standard language models of entities to minimize their overlap by discarding non-essential terms from them.

In the original parsimonious language model [136], the background model is explained by the estimation of the *collection model*, i.e., the model representing all the entities, similar to Equation 3.3. However, with respect to the hierarchical structure, and our goal in HSWLM for making the representations of entities separable from each other, we need to use the parsimonization technique in two different directions: 1) given ancestors of an entity, and 2) given its descendants. Hence, besides parsimonizing given a single parent entity in the upper layers,

Algorithm 3.1 Modified Model Parsimonization.

```
1: procedure PARSIMONIZE(e,B)
2:     for all term t in the vocabulary do
3:         $P(t|\theta_B) \xleftarrow{normalized} \sum_{b_i \in B} \left( P(t|\theta_{b_i}) \prod_{\substack{b_j \in B \\ j \neq i}} (1 - P(t|\theta_{b_j})) \right)$
4:         repeat
5:             E-Step: $P[t \in \mathcal{V}] \leftarrow P(t|\theta_e) \cdot \frac{\alpha P(t|\tilde{\theta}_e)}{\alpha P(t|\tilde{\theta}_e) + (1-\alpha) P(t|\theta_B)}$
6:             M-Step: $P(t|\tilde{\theta}_e) \leftarrow \frac{P[t \in \mathcal{V}]}{\sum_{t' \in \mathcal{V}} P[t' \in \mathcal{V}]}$
7:         until $\tilde{\theta}_t$ becomes stable
8:     end for
9: end procedure
```

as the background model, we need to be able to do parsimonization given multiple descendants in the lower layers. Algorithm 3.1 presents pseudo-code of the Expectation-Maximization algorithm that is employed in the modified model parsimonization procedure. In the equation in line 3 of the pseudo-code in Algorithm 3.1, B is the set of background entities —either one or multiple, and θ_{b_i} denotes the model of each background entity, b_i, which is estimated using MLE. In case of having a single ancestor node as the background model, this equation will be equal to Equation 3.3 and in case of having multiple descendants as background models, it results in Equation 3.8. In this procedure, in general, in the E-step, the probabilities of terms are adjusted repeatedly, and in the M-step, adjusted probabilities of terms are normalized to form a distribution. Another change in the modified version of model parsimonization, which practically makes no difference in the final estimation, is that in the E-step, instead of using $tf(t,e)$, we employ $p(t|\theta_e)$, where θ_e is the language model that represents entity e and initially it is estimated using MLE. This is because in the multi-layer hierarchies, there is more than one parsimonization pass for a particular entity and after the first round, we need to use the probability of terms estimated from the previous pass, not the raw information of their frequency.

Model parsimonization is an almost hyper-parameter free process. The only hyper-parameter is the standard smoothing parameter λ, which controls the level of parsimonization, so that the lower values of λ result in more parsimonious models. The iteration is repeated a fixed number of times or until the estimates do not change significantly anymore.

The pseudo-code of the overall procedure of estimating HSWLM is presented in Algorithm 3.2. Before the first round of the procedure, a standard estimation like maximum likelihood estimation is used to construct the initial representation for each entity in the hierarchy. Then, representations will be updated in an iterative process until all the estimated representations of entities become stable. In each

Algorithm 3.2 Procedure of estimating HSWLM.

```
1: procedure ESTIMATEHSWLMS
       Initialization:
2:     for all entity $e$ in the hierarchy do
3:         $\theta_e \leftarrow$ standard estimation for $e$ using MLE
4:     end for
5:     repeat
6:         SPECIFICATION()
7:         GENERALIZATION()
8:     until models do not change significantly anymore
9: end procedure
```

iteration, there are two main stages: a *Specification stage* and a *Generalization stage*. In these stages, language models of entities in the hierarchy are iteratively made "specific," by taking out terms explained at higher levels, and "general," by eliminating specific terms of lower layers, which results in representations that are both *horizontally* and *vertically* separable as it is described in Section 3.2.2.

In the specification stage, the goal is to eliminate the general terms of the language model of each entity so that the resulting language model demonstrates the entity's specific properties. To do so, the parsimonization method is used to parsimonize the language model of an entity given its ancestors, from the root of the hierarchy to its direct parent, as the background estimations. The order in the hierarchy is of crucial importance here. When a language model of an ancestor is considered as the background language model, it should demonstrate the "specific" properties of that ancestor. Due to this fact, it is important that before considering the language model of an entity as the background estimation, it has already passed the specification stage, and we have to move top-down. Pseudo-code of the recursive procedure of specification of entities' representations in the hierarchy is depicted in Algorithm 3.3.

In the generalization stage, the goal is to refine language models by removing terms that do not address the concepts in the level of abstraction of the entity's layer. To do so, again parsimonization is exploited but given descendants, which leads to the elimination of specific terms. Here also, before considering the representation of an entity as the background model, it should be already passed the generalization stage, so generalization moves bottom up. Algorithm 3.4 presents the pseudo-code for the recursive procedure of generalization of entities' language representations in the hierarchy. In the generalization step, the background models of descendants are supposed to be specific enough to show their extremely specific properties. Hence, generalization stages must be applied to the representations that are output of specification stages: specification should precede generalization, as shown in Algorithm 3.2 before.

Algorithm 3.3 Procedure of Specification.

```
1: procedure SPECIFICATION
2:     Queue ← all entities in breadth first order
3:     while Queue is not empty do
4:         $e$ ← Queue.pop()
5:         $l$ ← $e$.Depth()
6:         while $l > 0$ do
7:             $A$ ← $e$.GETANCESTOR($l$)
8:             PARSIMONIZE($e$,$A$)
9:             $l \leftarrow l - 1$
10:        end while
11:    end while
12: end procedure
```

▸ Function GETANCESTOR(l) gives the ancestor of entity e with l edges distance from it. Function PARSIMONIZE(e,B) parsimonizes θ_e given background models in B (Algorithm 3.1).

Algorithm 3.4 Procedure of Generalization.

```
1: procedure GENERALIZATION
2:     Stack ← all entities in breadth first order
3:     while Stack is not empty do
4:         $e$ ← Stack.pop()
5:         $l$ ← $e$.Height()
6:         while $l > 0$ do
7:             $D$ ← $e$.GETDECEDENTS($l$)
8:             PARSIMONIZE($e$,$D$)
9:             $l \leftarrow l - 1$
10:        end while
11:    end while
12: end procedure
```

▸ Function GETDECEDENTS(l) gives all the decedents of entity e with l edges distance from it. Function PARSIMONIZE(e,B) parsimonizes θ_e given background models in B (Algorithm 3.1).

3.4 HSWLM for Hierarchical Classification

Two-dimensional separability as the main property of HSWLM makes the learned representations of the entities in the hierarchy less sensitive to the structural changes, for instance, when the hierarchy evolves over time. In this section, we address our last research question of this chapter:

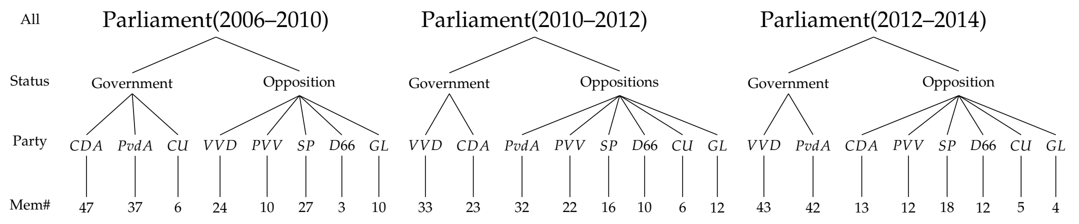


Figure 3.3: Composition of Dutch parliament in 3 periods. *VVD*: People's Party for Freedom and democracy, *PvdA*: Labour Party, *CDA*: Christian Democratic Appeal, *PVV*: Party for Freedom, *SP*: The Socialist Party, *D66*: Democrats 66, *GL*: Green-Left, *CU*: Christian-Union.

RQ-1.2.3 *How can separability of representations for hierarchical entities improve their transferability?*

Before evaluating the transferability of two-dimensionally separable representations over time, we intrinsically study the separability of the representations learned by HSWLM. We use parliamentary data as one of the interesting collections with hierarchically structured data that can evolve over time. The structure of the parliamentary hierarchy has been shown in Figure 3.1. First, we introduce the collection we have used, and then we analyze the quality of HSWLM on providing horizontal and vertical separability over the hierarchy.

3.4.1 Data Collection

We use the Dutch parliamentary data which forms a hierarchical structure that can evolve over time. The data are collected and annotated as part of *Political Mashup* project [221] to make semantically enriched parliamentary proceedings available as open data [191].

As a brief background, the Dutch parliament is a bicameral parliament that consists of a Senate and a House of Representatives. The House of Representatives is the main chamber of parliament, where discussion of proposed legislation and review of the government's actions takes place. The Dutch parliamentary system is a multi-party system, requiring a coalition of parties to form the government [61].

We use data from the parliament or the House of Representatives of the Netherlands. We have chosen three interesting periods of parliament, from March 2006 to April 2014, in which eight main parties have about 95% of seats in the parliament: People's Party for Freedom and Democracy, Labour Party, Christian Democratic Appeal, Party for Freedom, The Socialist Party, Democrats 66, Green-Left, and Christian-Union. The coalition in the first period is between a left-wing party and a centrist party, in the second period between a right-wing party and centrist party, and in the third, between a right-wing and left-wing party. Figure 3.3 shows the hierarchical structure of the Dutch parliament in these three different periods.

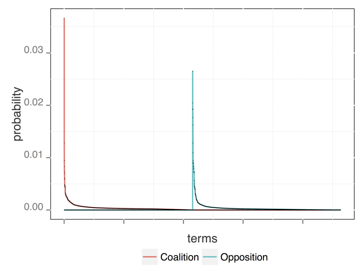
In order to learn representations for parliamentary entities, first of all, we prepare the data. In the proceedings, there are series of parliamentary speeches by different MPs following the debate structure. We invert the data matrix so that for each speaker we collect their speeches as a single document that reflects the features of that member. Then, for representing the internal entities in the parliament's hierarchy, we first consider members as the leaf entities and then concatenate all leaf documents below internal entities as a single document which textually represents them: first over parties, and then parties into government and opposition, etc. The whole corpus consists of 14.7 million terms from 240,501 speeches and contains 2.1 million unique terms. No stemming and no lemmatization is done on the data and also stop words, and common words are not removed in data preprocessing. After data preparation, we estimate SWLMs for all entities in the hierarchy as it is explained in Section 3.3.

3.4.2 Two-Dimensional Separability of HSWLM

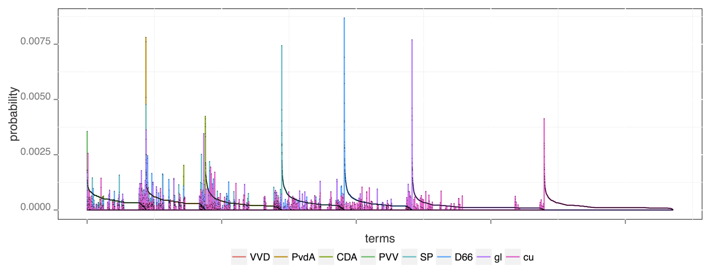
Here we investigate the ability of HSWLM to provide language models for hierarchical entities that are two-dimensionally separable. Based on the explained procedure of estimating HSWLM, the language models of entities in the hierarchy are repeatedly updated, so that the resulting in representations are both *horizontally* and *vertically* separable in the hierarchy. To assess this fact, we estimate HSWLM on the parliamentary data and look into the separability between entities in the same layer or in different layers.

Figures 3.4a and 3.4b illustrate the probability distributions over terms based on the estimated HSWLMs in the status and party layer, respectively. We sort the probability distribution based on the term weight of the first representation and plot the other representations in this exact order. As can be seen in the status layer, Figure 3.4a, the distributions over terms for government and opposition cover almost separated set of terms. Since in this layer these two entities are supposed to be against each other, a high level of separability can be expected. On the other hand, in the party layer, Figure 3.4b, it is possible that two parties share some ideological issues and consequently share some terms. So, in this layer, a complete separability of terms would not be practically possible for all the parties. Nevertheless, HSWLM, to some extent, provides horizontal separability in this layer.

Besides, we illustrate the horizontal separability of HSWLM of some pairs of parties. Figures 3.5a, 3.5b, and 3.5c show the separability of representations of two parties in three cases, respectively: 1) different statuses, 2) both in the status of opposition, 3) both in the status of government. It can be seen that in

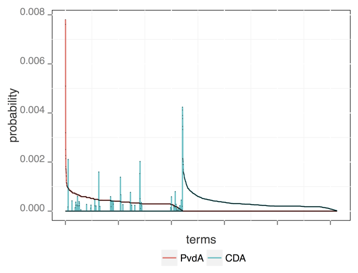


(a) HSWLM in the status layer.

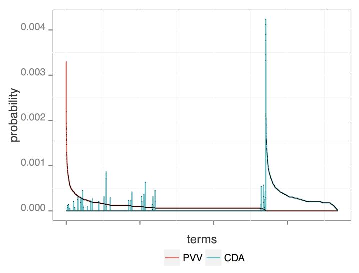


(b) HSWLM in the party layer.

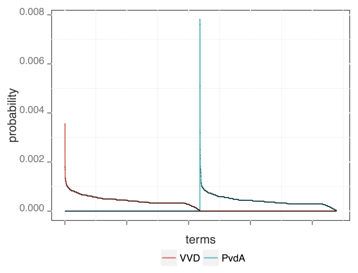
Figure 3.4: *Horizontal Separability*: probability distribution over terms based on hierarchical significant words language models in status layer and party layer.



(a) HSWLM of two parties in different statuses:CDA and PvdA.



(b) HSWLM of two parties in opposition: PVV and CDA.



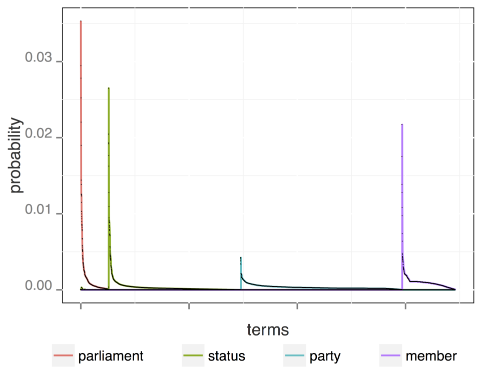
(c) HSWLM of two parties in government: VVD and PvdA.

Figure 3.5: *Horizontal Separability*: probability distribution over terms based on hierarchical significant words language models in party layer.

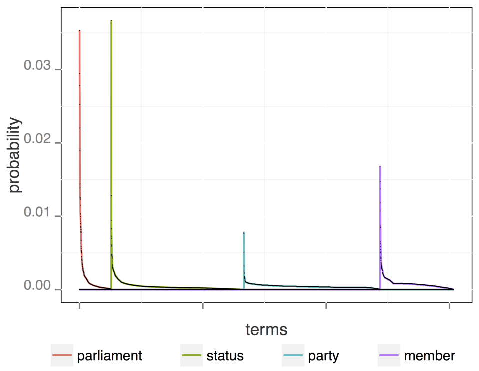
all cases of being in the same status or different status the estimated hierarchical significant words language models are separable. The interesting point is in Figure 3.5c, which the presents the representations of two government parties that are strongly separable. This is rooted in the fact that in this period there was an unusual coalition government consisting of a right-wing and a left-wing party. So, although they have an agreement in the status layer, their representation is highly separable in terms of having opposite spectrums in party layer.

In order to illustrate the vertical separability of HSWLM, we choose two different branches in the hierarchy: one from the leader of one of the opposition parties to the root, and the other from the leader of one of the government parties to the root. Figure 3.6a and 3.6b show probability distributions over words based on HSWLM of all entities in these two branches. They show that using HSWLM, we can decompose the distribution over all terms into highly separable distributions, each one representing the language usage related to the meaning behind the layer of the entity in the hierarchy.

The two-dimensional separation property of HSWLM in the hierarchy is essentially due to the parsimonization effect in two directions. Intuitively, the horizontal separability is mainly the result of the specification stage. For example, when an entity is parsimonized given its direct parent, since the data in its parent is formed by pooling the data from the entity and its siblings, parsi-



(a) HSWLM of S. van Haersma Buma (as the member of parliament - Leader of CDA), CDA (as the party), Opposition (as the status), and the Parliament.



(b) HSWLM of D. Samson (as the member of parliament - Leader of PvdA), PvdA (as the party), Government (as the status), and the Parliament.

Figure 3.6: *Vertical Separability*: probability distribution over terms in different layers based on hierarchical significant words language models in complete paths from the root to the terminal entities in the hierarchy.

monization makes the representation of the entity separable from its siblings, which provide *horizontal separation* in the resulting language models. On the other hand, vertical separability is mainly due to the generalization stage (and implicit specification). For example, when an entity is parsimonized given its children, since they are specified already, parsimonization gets rid of the specific terms of the lower layer from the entity's representation.

3.4.3 Separability for Transferability

As an extrinsic evaluation of the hierarchical significant words language models, we investigate the effectiveness of the learned representations in a classification task in the parliamentary dataset with an evolving hierarchical structure. The task is to predict either the party that a member of the parliament belongs to, or the status of the member's party, having all the speeches given by that member in a period, as well as all the speeches given by the members of all parties in a different period of parliament.

In the parliament, the composition of parties and statuses changes over different periods (Figure 3.3) and hence the speeches related to different entities can vary dramatically. Due to this fact, cross period classification is notoriously challenging [139, 317]. We show that representing entities with HSWLM tackles the problem of having non-stable representations when the composition of parliament evolves during the time, by capturing the essence of language models of parliamentary entities at aggregate levels.

We use SVM as the base classifier.We trained a standard SVM on raw text as well as a SVM that uses the probabilities of terms in HSWLM as the weights of features. Using the probabilities estimated by HSWLM as weights for features

Table 3.1: Results on the task of status classification.

(a) Accuracy of the SVM classifier.

	Period	Test			
		2006–10	*2010–12*	*2012–14*	*All*
Train	*2006–10*	84.14	68.83	87.24	-
	2010–12	68.29	78.57	87.91	-
	2012–14	68.90	75.97	88.59	-
	All	-	-	-	79.87

(b) Accuracy of the SVM_{HSWLM} classifier.

	Period	Test			
		2006-10	2010-12	2012-14	All
Train	2006-10	82.32	80.51	89.29	-
	2010-12	79.87	74.66	88.58	-
	2012-14	78.65	77.27	93.28	-
	All	-	-	-	86.98

Table 3.2: Results on the task of party classification.

(a) Accuracy of the SVM classifier.

	Period	Test			
		2006–10	*2010–12*	*2012–14*	*All*
Train	*2006–10*	47.56	29.22	26.84	-
	2010–12	29.87	40.90	35.57	-
	2012–14	31.09	30.51	44.96	-
	All	-	-	-	39.18

(b) Accuracy of the SVM_{HSWLM} classifier.

	Period	Test			
		2006-10	2010-12	2012-14	All
Train	2006-10	44.51	46.10	43.62	-
	2010-12	40.85	40.25	39.59	-
	2012-14	40.24	38.96	42.28	-
	All	-	-	-	49.94

can be considered as a feature selection approach that filters out features that are not essential in accordance to the hierarchical position of entities and make the data representation more robust by taking out non-stable terms.[1] We have employed conventional 5-fold cross validation for training and testing and to maintain comparability, we have used the same split for folding in all the experiments.

Tables 3.1b and 3.2b show the performance of employing SWLM on status and party classification, respectively. Tables 3.1a and 3.2a indicate the results of SVM classifier on status and party classification respectively. Comparing the results in Tables 3.1a and 3.1b, we see that the accuracy of SVM in within period experiments is sometimes slightly better, but in cross period experiments, the classifier that uses SWLM of statuses achieves better results. This is also observed in the results in Table 3.2b compared to the results in Table 3.2a.

For party classification, employing SWLM results in a significant improvement over the baseline. Hirst et al. [139] discuss that since the status of members in parliament, compared to their party, has more effect on the content of their speeches, classifiers tend to pick features related to the status, not the party ideologies. So, SVM performs very well in terms of accuracy in the within-period experiments, but this performance is due to the separability of parties due to their status. Hence, changing the status in cross period experiments, using the learned representation on other periods fails to predict the party, so the accuracies drop. This is exactly the point where our proposed method kicks in. Since for each party, the SWLM is less affected by the status of the party in that period, the representation remains valid even when the status is

[1]We have also tried SVM along with a feature selection methods [29, 101] that uses the Information Gain (IG) to select features as a baselines and reported the results in [68].

changed. In other words, eliminating the effect of the status layer in the party representation in the specification stage ensures that the party language model captures the essential terms related to the party ideology, not its status. Thereby, it is a stable representation that is transferable over time. We conducted a one-tailed t-test on the results. In both party and status classification, in all cases where SWLM performs better than the SVM, the improvement is statistically significant (p-value < 0.005).

To get a better intuition of the procedure of estimating SWLM, consider the hierarchical relations of Dutch parliaments in the period of *2006–2010*, which is depicted in Figure 3.3. Assume that the goal is modeling language usage of "Christian-Union (CU)" as an entity in the party layer. In the speeches from the members of this party, words like "*Chairman*" or "*Agree*" might occur repeatedly. However, they are not a good point of reference for the party's ideological language usage. In the procedure of estimating SWLM of the "Christian-Union," these words are removed from the initial estimated standard language model in the specification stages, since "*Chairman*" is a general term in the parliamentary domain and is only able to explain the root entity and "*Agree*" is somehow an indicator of language usage of all the "Government" parties. On the other side, assume that the goal is to model language usage of "Government" as an entity in the status layer. Speeches from "Christian-Union" members, which are also counted as "Government" members, may contain words like "*Bible*" or "*Charity*." It is trivial that involving these party-specific words in the learned representation for the "Government" in an individual period hurts the comprehensiveness. In the procedure of estimating SWLM for the "Government," in the generalization stages, these words are discarded from the representation. This way, the "Government" representation does not lose its validity on other periods where the "Christian-Union" is not in a Government party.

As another indicator of the effectiveness of SWLM, it outperforms the SVM bringing all the data together from three different periods in both party and status classification. This is because it gets the chance of having richer training data, which leads to more precise models. While in SVM, changes in the parliamentary composition make speeches diverse, as a result of which it is not able to learn a concrete model.

3.4.4 Invariance of the Representations

As an intrinsic evaluation of the models, we evaluate the invariance of representations learned by our model over different periods—how similar are representations of a particular entity in the hierarchy when trained on data from different periods.

To assess this, we use the diversity of entities' representations in different

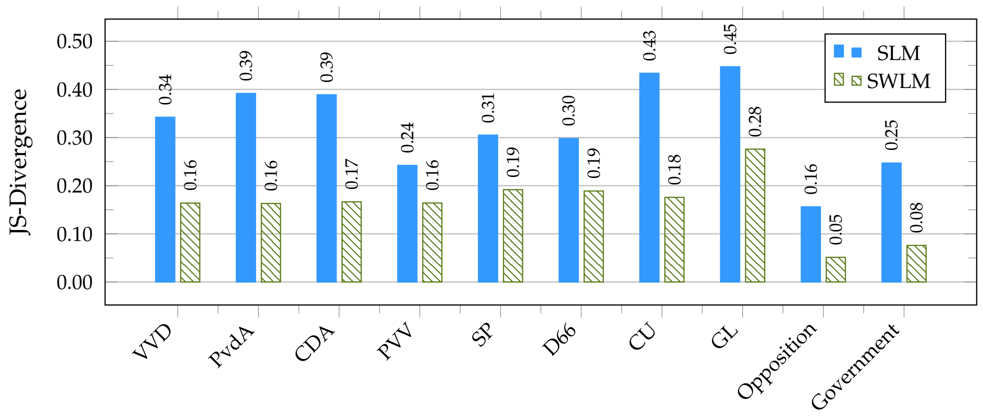


Figure 3.7: Average of JS-Divergence of standard language models and SWLMs for parliamentary entities in three different periods.

periods to measure their (in)variance over time. First, all HSWLMs from different periods of each party and each status are smoothed using Jelinek-Mercer smoothing [330] considering all parliamentary speeches in the corresponding period as the background collection and with the same value of the smoothing parameter. Then, we use the Jensen-Shannon divergence as the diversity metric to measure dissimilarities between each two HSWLMs learned from different periods and then calculate the average of values for each entity. As the baseline, the same calculation is done for the standard language models of entities, i.e., language models estimated using maximum likelihood estimation. Figure 3.7 shows the diversity of representations in different periods. As can be seen, in all entities in both party and status layers, the diversity of HSWLM of different periods is lower than the diversity of standard language models, which shows that the estimated HSWLMs are more invariant over different periods.

In order to better understand the results in the previous section, we zoomed in on the period of 2010–2012 and 2012–2014 and looked into the confusion matrices of cross period experiments and observed that most of the errors made by SVM are misclassifying members of CDA to PvdA and vice versa. These are the two parties whose statuses have been changed in these periods.

We investigate representations of these two parties to understand how separation in the feature representation affects the performance of cross period classification. To do so, for each of these two classes, in each period, we extract three probability distributions on terms indicating their importance based on different weighting approaches: 1) Term Frequency (used as feature weights in SVM) and 2) probability of terms in HSWLM (used as feature weights in SVM_{HSWLM}). Then, as a metric to measure separability of features, we use the Jensen-Shannon divergence to calculate diversity of probability distributions in three cases: 1) Different Parties in the Same Period, 2) Same Party in Different Periods 3) Different Parties in Different Periods. To avoid the effect of the num-

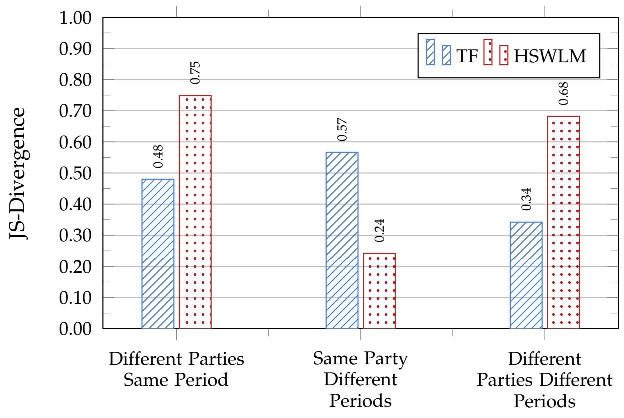


Figure 3.8: Average diversity of the representation of features of CDA and PvdA in different situations.

ber of features on the value of divergence, we take the top-500 highest scoring terms of each of the weighting methods as their fixed length representatives. Figure 3.8 shows the average diversity of distributions in each of the three cases for each of the three weighting methods.

As expected, the diversity of features for different parties in a single period is high for both methods. However, when we calculate the diversity of features for a single party in different periods, feature representations are different in TF, which causes false negative errors in the classification of these two parties. When using representations using TF, we get similar representations for different parties in different periods, probably due to the fact that they share status. This can lead to false positive errors in the classification.

Considering these observations together reveals that SVM learns representations on the basis of features that are indicators of issues related to the status of parties, since they are the most discriminating terms considering one period and in within period experiments, the performance of SVM is due to the separability of parties based on their statuses. Hence, after changing the status in the cross period experiments, the trained model of the previous period generated by SVM fails to predict the accurate party. In the same way, the status classifier is affected by different parties forming a government in different periods, leading to lower accuracies.

This is exactly the point where the strengths of HSWLM kick in. In fact, two-dimensional separability in the feature representation enables SVM_{HSWLM} to tackle the problem of having non-stable features in the representation when the status of a party changes over time. In other words, eliminating the effect of the status layer in the party representation, which is the result of the horizontal separation, ensures that the party representation captures the terms related to the party ideology, not its status. Thereby, SVM_{HSWLM} is not only effective in within period classification, but also the models learned on the data from one period remain valid when the statuses of parties change in other periods.

3.5 Related Work

This section briefly discusses the separation property in the related domains and reviews principles in information retrieval and text classification that are associated with the concept of separability. In addition, some research on classification and topic modeling of hierarchical texts is discussed.

Separability is a property that makes the data representation sufficiently rich to distinguish instances and consequently enables autonomous systems to easily interpret the data [174]. For instance, in the classification task, classifiers learn more accurate data boundaries when they are provided with separable representations of data from different classes [175]. The importance of separability in classifiers has led to the fact that making data separable becomes part of the classification. As the most familiar instances, SVM transform the data into a new space where they are linearly separable [36].

Separation is also a pivoting concept in information retrieval (IR). Separating relevant from non-relevant documents is a fundamental issue in this domain [169, 231, 245]. In IR, separation plays a more important role when instead of giving a ranked list, a decision should be made about the relevancy of documents, for example in the information filtering task [174]. As another instance, in the task of relevance feedback, there are some efforts on estimating a distinctive representation for relevant documents so that it reflects not only their similarity, but also their difference from the whole collection, i.e., what makes them stand out or separated [136, 263, 329].

In this chapter, we address the separation property in textual data that is organizable in a hierarchical structure. In a hierarchy, due to the existence of dependencies between entities, estimating separable representations is a complex task. There is a range of work on the problem of hierarchical text classification [246, 268], which tried to model hierarchical text-based entities. McCallum et al. [193] proposed a method for learning representation for an entity in the hierarchy which tackles the problem of data sparseness in lower layer entities. They used a shrinkage estimator to smooth the representation of each leaf entity with the representation of its ancestors to make them more reliable. There is also similar research on XML data processing, as hierarchically structured data, which tries to incorporate evidence from other layers as the context through mixing each element language models by its parent's models [206, 258].

Recently, Song and Roth [262] tackled the problem of representing hierarchical entities with a lack of training data for the task of hierarchical classification. In their work, given a collection of instances and a set of hierarchical labels, they tried to embed all entities in a semantic space, then they construct a semantic representation for them to be able to compute meaningful semantic similarity between them.

Zhou et al. [333] proposed a method that directly tackles the difficulty of modeling similar entities at lower levels of the hierarchy. They used regularization so that the representation of lower level entities have the same general properties as their ancestors, in addition to some more specific properties. Although these methods tried to learn representation for hierarchical texts, their concerns were not making the representations separable. Instead, they mostly addressed the problem of training data sparseness [120, 193, 262] or presented techniques for handling large scale data [111, 120, 207, 311].

In terms of modeling hierarchical entities, Kim et al. [159] used Hierarchical Dirichlet Processes (HDPs) [275] to learn representations for entities in the hierarchies using their own representations as well as representations of their ancestors. Also, Zavitsanos et al. [327] used HDPs to learn a representation for entities in a hierarchy employing representations of their descendants. This research tries to bring out precise topic models using the structure of the hierarchy, but they do not aim to estimate separable representations.

As we discussed in Section 3.4.3, our proposed approach can be employed as a feature selection method for text classification. Prior research on feature selection for textual information [101, 104] tried to improve classification accuracy or computational efficiency, while our method aims to provide a separable representation of data that helps train a transferable model. Apart from considering the hierarchical structure, our goals also differ from prior research on the transferability of models. For instance, research on constructing dynamic models for data streams [25, 315] first discovered the topics from data and then tried to efficiently update the models as data changes over the time, while our method aims to identify tiny precise representations that are more robust and remain valid over time. Research on domain adaptation [42, 310] also tried to tackle the problem of missing features when very different vocabularies are used in test and training data. This differs from our approach considering the hierarchical relations, as we aim to estimate separable representations that are robust against changes in the structure of entities relations, rather than changes in the corpus vocabulary.

3.6 Conclusion

The wish to learn conceptually accurate representations of data with a structure consisting of multiple layers, or a hierarchy, prompts us to analyze the data at different abstraction levels. However, this requires the ability to estimate separable representations for hierarchical entities that capture their essential features taking into account their relative position in the hierarchy.

In this chapter, we focused on addressing **RQ-1.2**: *"How to learn separable representations for hierarchically structured entities that are less sensitive to structural changes in the data and more transferable across time?"*. We demonstrated that based on the ranking and classification principles, the *separation property* in the data representation is a desirable foundational property that leads to separability of scores and consequently improves the accuracy of classifiers' decisions, which addressed **RQ-1.2.1**. We stated this as the "Strong Separation Principle" for optimizing expected effectiveness of classifiers.

We showed that in order to have horizontally and vertically separable representations for hierarchically structured data, they should capture all, and only, the essential features of the entities taking their position in the hierarchy into account. Based on this, to address **RQ-1.2.2**, we introduced hierarchical significant words language models for estimating separable representations for hierarchical entities. We studied HSWLMs and demonstrated that the offer separable distributions over terms for different entities both in the case of being in the same layer or in different layers. To study **RQ-1.2.3**, we evaluated the performance of classification over time using separable representations of data and showed that separability makes the representation more robust and transferable over time by filtering out non-essential and non-stable feature.

In Part I of the thesis, we have focused on addressing **RQ-1**: *"How to design learning algorithms that can learn from weakly annotated samples, while generalizing over the imperfection in their labels?,"* by studying how incorporating some prior knowledge can help improve the robustness of the outcome of the learning process in noisy and variable environments. We have shown that taking the structure of the data, which is in fact a form of inductive bias, into account can help to learn effective representations. Next, in Part II, we will investigate how we can change the learning process to make it more robust against noise in the weakly annotated labels.

Part II

Learning with Weak Supervision

The unprecedented success of data-driven approaches in machine learning has turned data into a first-class citizen in machine learning. Most of the times, the more data you have, the more accurate your model will be [121, 269] and it is more crucial to provide massive amounts of training data when the models become more deep and complex. Collecting such training sets by hand is often infeasible due to the time and expense of labeling data. Besides, hand-labeled training sets are static and we might need complete relabeling, for instance, when the modeling goals changes. Thus moving beyond fully supervised learning, like adopting weak supervision with the hope to overcome the poverty of stimulus, is a key direction in machine learning research.

Humans can learn from weak and inconsistent signals[48]. However, it seems difficult to build fault-tolerant machine learning systems that learn, even though imperfect signals can contain a great deal of valid information. Given the fact that only a tiny portion of real word applications operate under perfect conditions, an essential aspect of any practical learning algorithm is the need to learn from inconsistent data provided by different sensors, noisy or weak supervision, and even when crucial information is missing from the supervision signal.

In Part II of this thesis, we address the following research question:

> **RQ-2** *How to design learning algorithms that can learn from weakly annotated samples, while generalizing over the imperfection in their labels?*

The imperfect samples can come from labels provided by non-expert crowd workers, be the output of other models that are weaker (for instance, with low accuracy or coverage), biased, or models trained on data from different related domains. In this part, we aim to study how we can provide supervision signals for machine learning systems by labeling training data programmatically instead of labeling by hand [226]. Then, given a vast amount of programmatically generated labeled data, and maybe a small set of samples with true labels, we discuss how to design neural networks that leverage the full capacity of the information in the data and go beyond the imperfection of weakly annotated data.

In the first chapter of this part, Chapter 4, we address the following research question:

> **RQ-2.1** *How can we train neural networks using programmatically generated pseudo-labels as a weak supervision signal, in a way that they exhibit superior generalization capabilities?*

In this chapter, we propose to train a neural ranking model using weak labels that are obtained automatically without human annotators or any external resources (e.g., click data). We train a set of simple yet effective neural ranking models and study their effectiveness under various learning scenarios, i.e., point-wise and pair-wise, different objective functions, and using different input representations, from using a set of engineered features to encoding query/document using word embeddings [84]. We also discuss how privacy preserving approaches can benefit from models that are capable of learning from weak signals, where instead of labels from the original sensitive training data a noisy version is provided [71].

Then, in the second chapter of this part, Chapter 5, we focus on the following research question:

> **RQ-2.2** *Given a large set of weakly annotated samples and a small set of samples with high-quality labels, how can we best leverage the capacity of information in these sets to train a neural network?*

In this chapter we introduce *Learning with Controlled Weak Supervision (CWS)* [82, 83] and *Fidelity Weighted Learning (FWL)* [79, 80], two semi-supervised approaches for training neural networks, where we have a large set of data with weak labels and a small amount of data with true labels. In CWS we train two neural networks in a meta-learning setup: a target network, the learner and a confidence network, the meta-learner. The target network is optimized to perform a given task and is trained using a large set of unlabeled data that is weakly annotated. We propose to control the magnitude of the gradient updates to the target network using the scores provided by the second confidence network, which is trained on a small amount of supervised data. Thus we avoid that the weight updates computed from noisy labels harm the quality of the target network model. FWL is a student-teacher approach in which we modulate the parameter updates to a *student* network (trained on the task we care about) on a per-sample basis according to the posterior confidence of its label-quality estimated by a *teacher* (who has access to the high-quality labels).

We show that we can train a neural ranker using a heuristic labeling function as weak supervision signal and go beyond the performance of this weak annotator, merely by choosing the right architecture and objective functions, and discuss how this can benefit learning in a privacy-preserving setup. Given a semi-supervised setup, we apply our introduced methods, CWS and FWL, to a range of language understanding tasks and empirically verify that they improve over semi-supervised alternatives and speeds up the training process.

4 Learning from Pseudo-Labels

In many applications with scarce training data, we can provide supervision signals for learning algorithms by pseudo-labeling the data programmatically, rather by hand. This way, we can generate a much larger training set with low cost. But it requires designing algorithms that are capable of learning from weakly annotated labels and of going beyond the imperfection of pseudo-labels.

4.1 Introduction

Neural networks are making great progress in many tasks in computer vision [165], natural language processing [56], and information retrieval [301]. However, these models are data hungry and their performance is strongly correlated with the amount of available labeled data, which is not always readily available and can be expensive to obtain.

Looking into the research done in this area, most of it targets stable benchmark tasks where standard large-enough datasets exist to train neural networks. However, the labeled data becomes a scarce commodity when we stray slightly from these standard benchmark tasks toward the realm of real-world applications. In this chapter, we focus on one of our research questions:

This chapter is based on [71, 81, 84].

> **RQ-2.1** *How can we train neural networks using programmatically generated pseudo-labels as a weak supervision signal, in a way that they exhibit superior generalization capabilities?*

We aim to study we can provide supervising signal for machine learning systems, by labeling training data programmatically instead of labeling by hand. Then, given a vast amount of programmatically generated labeled data, we discuss how to design neural networks that can go beyond the imperfection of weakly annotated data. We also study how the ability to learn from noisy signals can lead to better performance when we have intentionally added noise to the training signals in a privacy-preserving training setup.

In this chapter, we mainly target the *ranking task*, as one of the core IR problems, where despite the advances of neural network-based methods in many other related tasks like reading comprehension, there has been a little progress, mainly due to the lack of a large-scale public dataset with query-document pairs labeled by relevance.

We propose to use a heuristic-based ranking method to generate pseudo-labels for a large set of unlabeled query-document pairs to train a neural ranking model given these pseudo-labels as a sort of weak annotations. We try different architectures, in terms of different objectives and different input representations and study how they learn the ranking task in weak supervision setup.

Interestingly, we observe that using just training data that are annotated by a heuristic unsupervised model as the weak annotator, we can outperform that weak annotator on the test data. Based on our analysis, the achieved performance is generally due to three main factors: First, defining an objective function that aims to learn the ranking instead of calibrated scoring to relax the network from fitting to the imperfection of weakly supervised training data. Second, letting the neural networks learn optimal query/document representations instead of feeding them with a representation based on predefined features. This is a key requirement to maximize the benefits from deep learning models with weak supervision as it enables them to generalize better. Third and last, the weak supervision setting makes it possible to train the network on a massive amount of training data, which is crucial for learning representations.

We further thoroughly analyze the behavior of models to understand what they learn, what kinda of the relationship is among different models, and how much training data is needed to go beyond the weak supervision signal. We also examine the scenario of using the network trained on a weak supervision signal as a pre-training step. We demonstrate that, in the ranking problem, the performance of deep neural networks trained on a limited amount of supervised data significantly improves when they are initialized from a model pre-trained on weakly labeled data.

Finally, we study how a neural ranking model that learns from weak/noisy signals can be effectively employed in a setup in which noise is intentionally added to the training signal to preserve privacy.

4.1.1 Detailed Research Questions

We break down our main research question in this chapter into three concrete research questions:

> **RQ-2.1.1** *Can labels from an unsupervised heuristic-based model be used as programmatically generated weak supervision signal to train an effective neural network?*
>
> **RQ-2.1.2** *What setup in terms of input representation and learning objective is most suitable for a neural ranker when training on programmatically generated labeled data?*
>
> **RQ-2.1.3** *How can learning from weak supervision signals help to preserve privacy while training neural networks on sensitive data?*

In the following sections, we will address these research questions.

4.2 Weakly Supervised Neural Rankers

Despite the promising performance from neural networks on many language understanding tasks [30, 134], ranking has remained a challenging problem. Besides the inherent difficulty of "assessing relevance," the lack of availability of public large-scale datasets that consist of query-document pairs annotated with relevance labels, makes it difficult to advance data hungry models for this task.

Thus, the ranking task is one of the areas that requires solutions that enable us training neural networks, where there is little to no labeled data is available. One of the main ideas to tackle this problem is to make use of weak human supervision or weakly labeled data, as it is much cheaper to collect or use readily available at much larger-scale. This section focuses on addressing the following questions:

> **RQ-2.1.1** *Can labels from an unsupervised heuristic-based model be used as programmatically generated weak supervision signal to train an effective neural network?*

We propose to pseudo-label a large set of unlabeled data using an unsupervised method and train a neural ranker using these "weak" or "noisy" labels. Given this setup, we examine various neural ranking models with different ranking architectures and objectives, i.e., point-wise and pair-wise, as well as different input representations, from encoding query-document pairs into dense / sparse vectors to learning query / document embedding representations.

Our results have broad impact as the proposal to use unsupervised traditional methods as weak supervision signals and is applicable to a variety of IR tasks, such as filtering or classification, without the need for supervised data. More generally, our approach unifies the classic IR models with currently emerging data-driven approaches in an elegant way.

4.2.1 Pseudo-Labeling Unlabeled Data

We use the idea of "Pseudo-Labeling" and propose to leverage a classic unsupervised IR model to annotate a large amount of unlabeled data and infer weak labels and use this signal to train supervised models as if we had the ground truth labels. Since the data is generated programmatically, we can generate billions of training samples with almost no cost.[1]

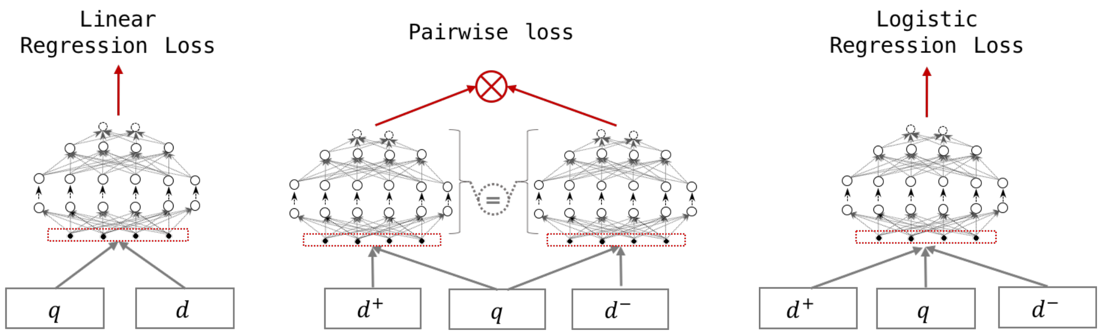
We focus on query-dependent ranking as a core IR task. To this aim, we take a well-performing existing unsupervised retrieval model, such as BM25. This model plays the role of "pseudo-labeler" in our learning scenario. In more detail, given a target collection and a large set of training queries (without relevance judgments), we make use of the pseudo-labeler to rank/score the documents for each query in the training query set. The goal is to train a ranking model given the scores/ranking generated by the pseudo-labeler as a weak supervision signal.

In the following, we describe different neural architectures in detail and finally investigate their effectiveness when trained on weakly annotated data.

4.2.2 Neural Ranking Architectures

In this section, we introduce three different neural ranking models that are trained based on different "objectives." We describe the architecture of the base neural network shared by these models. We further discuss the three different "input layers" used in our neural rankers to encode information of given query-document pairs.

[1]Weak supervision for training a ranker may refer to using click-through data. Here, we assume that no external information, e.g., search logs, is available.



(a) Score model (b) Rank model (c) RankProb model

Figure 4.1: Different ranking architectures.

We define three different ranking models, one point-wise and two pair-wise models:

Score Model

This architecture models a point-wise ranking model that learns to predict retrieval scores for query-document pairs. More formally, the goal in this architecture is to learn a *scoring function* $\mathcal{S}(q, d; \theta)$ that determines the retrieval score of document d for query q, given a set of model parameters θ. In the training stage, we are given a training set comprising of training samples, each of which is a triple $x = (q, d, s_{q,d})$, where q is a query from training query set Q, d represents a retrieved document for the query q, and $s_{q,d}$ is the relevance score (calculated by a weak supervisor), which is acquired using a retrieval scoring function in our setup. We consider the mean squared error as the loss function for a given batch of training samples:

$$\mathcal{L}(b;\theta) = \frac{1}{|b|}\sum_{i=1}^{|b|}(\mathcal{S}(\{q,d\}_i;\theta) - s_{\{q,d\}_i})^2 \tag{4.1}$$

where $\{q, d\}_i$ denotes the query and the corresponding retrieved document in the i^{th} training sample, i.e., x_i in the batch b. The conceptual architecture of the model is illustrated in Figure 4.1a.

Rank Model

In this model, similar to the previous one, the goal is to learn a scoring function $\mathcal{S}(q, d; \theta)$ for a given pair of query q and document d with the set of model parameters θ. However, unlike the previous model, we do not aim to learn a calibrated scoring function. In this model, as is depicted in Figure 4.1b, we use a pair-wise scenario during training in which we have two point-wise networks that share parameters and we update their parameters to minimize a pair-wise loss. Each training sample has five elements: $x = (q, d_1, d_2, s_{q,d_1}, s_{q,d_2})$. During the inference, we treat the trained model as a point-wise scoring function to score query-document pairs.

We have tried different pair-wise loss functions and empirically found that the model learned based on the hinge loss (max-margin loss function) performs better than the others. Hinge loss is a linear loss that penalizes examples that violate the margin constraint. It is widely used in various learning to rank algorithms, such as Ranking SVM [133]. The hinge loss function for a batch of training samples is defined as follows:

$$\begin{aligned}\mathcal{L}(b;\theta) = \frac{1}{|b|}\sum_{i=1}^{|b|} \max\big\{0, \varepsilon - \operatorname{sign}(s_{\{q,d_1\}_i} - s_{\{q,d_2\}_i}) \\ (\mathcal{S}(\{q,d_1\}_i;\theta) - \mathcal{S}(\{q,d_2\}_i;\theta))\big\},\end{aligned} \tag{4.2}$$

where ε is the parameter determining the margin of the hinge loss. We found that as we compress the outputs to the range of $[-1,1]$, $\varepsilon = 1$ works well as the margin for the hinge loss function.

RankProb Model

The third architecture is based on a pair-wise scenario during both training and inference (Figure 4.1c). This model learns a *ranking function* $\mathcal{R}(q, d_1, d_2; \theta)$ that predicts the probability of document d_1 to be ranked higher than d_2 given q. Similar to the *rank* model, each training sample has five elements: $x = (q, d_1, d_2, s_{q,d_1}, s_{q,d_2})$. For a given batch of training samples, we define our loss function based on cross-entropy as follows:

$$\begin{aligned}\mathcal{L}(b;\theta) = -\frac{1}{|b|}\sum_{i=1}^{|b|} P_{\{q,d_1,d_2\}_i} \log(\mathcal{R}(\{q,d_1,d_2\}_i;\theta)) \\ + (1 - P_{\{q,d_1,d_2\}_i}) \log(1 - \mathcal{R}(\{q,d_1,d_2\}_i;\theta)),\end{aligned} \tag{4.3}$$

where $P_{\{q,d_1,d_2\}_i}$ is the probability of document d_1 being ranked higher than d_2, based on the scores obtained from training sample x_i:

$$P_{\{q,d_1,d_2\}_i} = \frac{s_{\{q,d_1\}_i}}{s_{\{q,d_1\}_i} + s_{\{q,d_2\}_i}}. \tag{4.4}$$

A similar loss function has previously been used in RankNet [35]. It is notable that at inference time we need a scalar score for each document. Therefore, we need to turn the model's pair-wise predictions into a score per document. To do so, for each document, we calculate the average of predictions against all other candidate documents, which has $O(n^2)$ time complexity and is not practical in real-world applications. There are some approximations the could be applicable to decrease the time complexity at inference time [300].

As shown in Figure 4.1, all the described ranking architectures share a neural network module. In all these models, we opted for a simple feed-forward neural network that is composed of: input layer z_0, $l-1$ hidden layers, and the output layer z_l. The input layer z_0 provides a mapping ψ to encode the input query and document(s) into a fixed-length vector. The exact specification of the input representation feature function ψ is given in the next subsection. Each hidden layer z_i is a fully-connected layer that computes the following transformation:

$$z_i = \alpha(W_i.z_{i-1} + b_i);\ 1 < i < l-1, \tag{4.5}$$

where W_i and b_i respectively denote the weight matrix and the bias term corresponding to the i^{th} hidden layer, and $\alpha(.)$ is the activation function. We use the rectifier linear unit $ReLU(x) = \max(0, x)$ as the activation function, which is a common choice in the deep learning literature [170]. The output layer z_l is a fully-connected layer with a single continuous output. The activation function for the output layer depends on the ranking architecture that we use. For the *score* model architecture, we empirically found that a linear activation function works best, while *tanh* and the sigmoid functions are used for the *rank* model and *rankprob* model, respectively.

Furthermore, to prevent feature co-adaptation, we use dropout as the regularization technique in all the models. Dropout sets a portion of hidden units to zero during the forward phase when computing the activations, which prevents overfitting.

4.2.3 Representing Inputs

We explore three definitions of the input layer representation z_0 captured by a feature function ψ that maps the input into a fixed-size vector which is further fed into the fully connected layers: (i) a conventional dense feature vector representation that contains various statistics describing the input query-document pair, (ii) a sparse vector containing bag-of-words representation, and (iii) bag-of-embeddings averaged with learned weights. These input representations define how much capacity is given to the network to extract a discriminative signal from the training data and thus result in different generalization behavior of the networks. It is noteworthy that the input representation of the networks in the *score* model and *rank* model is defined for a pair of a query and a document, while the network in the *rankprob* model needs to be fed by a triple of the query, the first document, and the second document.

Dense Vector Representation (Dense)

In this setting, we build a dense feature vector composed of features used by traditional IR methods, e.g., BM25. The goal here is to let the network fit the function described by the BM25 formula when it receives exactly the same inputs. In more detail, our input vector is a concatenation ($||$) of the following inputs: total number of documents in the collection (i.e., N), average length of documents in the collection (i.e., $avg(l_d)_D$), document length (i.e., l_d), frequency of each query term t_i in the document (i.e., $tf(t_i, d)$), and document frequency of each query term (i.e., $df(t_i)$). Therefore, for the point-wise setting, we have the following input vector:

$$\psi(q,d) = [N||avg(l_d)_D||l_d||\{df(t_i)||tf(t_i,d)\}_{1\leq i \leq k}], \tag{4.6}$$

where k is set to a fixed value (5 in our experiments). We truncate longer queries and do zero padding for shorter queries. For the networks in the *rankprob* model, we consider a similar function with additional elements: the length of the second document and the frequency of query terms in the second document.

Sparse Vector Representation (Sparse)

Next, we move away from a fully featurized representation that contains only aggregated statistics and let the network performs feature extraction for us. In particular, we build a bag-of-words representation by extracting term frequency vectors of query (tfv_q), document (tfv_d), and the collection (tfv_c) and feed the network with concatenation of these three vectors. For the point-wise setting, we have the following input vector:

$$\psi(q,d) = [tfv_c||tfv_q||tfv_d] \tag{4.7}$$

For the network in the *rankprob* model, we have a similar input vector with both tfv_{d_1} and tfv_{d_2}. Hence, the size of the input layer is $3 \times$ *vocab size* in the point-wise setting, and $4 \times$ *vocab size* in the pair-wise setting.

Embedding Vector Representation (Embed)

The major weakness of the previous input representation is that words are treated as discrete units, hence prohibiting the network from performing soft matching between semantically similar words in queries and documents. In this input representation paradigm, we rely on word embeddings to obtain more powerful representations of queries and documents that could bridge the lexical chasm. The representation function ψ consists of three components: an embedding function $\mathcal{E} : \mathcal{V} \to \mathbb{R}^m$ (where $\mathcal{V}$ denotes the vocabulary set and m is the embedding dimension), a weighting function $\mathcal{W} : \mathcal{V} \to \mathbb{R}$, and a compositionality function $\odot : (\mathbb{R}^m, \mathbb{R}^n) \to \mathbb{R}^m$. More formally, the function ψ for the point-wise setting is defined as:

$$\psi(q,d) = [\odot_{i=1}^{|q|}(\mathcal{E}(t_i^q), \mathcal{W}(t_i^q)) || \odot_{i=1}^{|d|} (\mathcal{E}(t_i^d), \mathcal{W}(t_i^d))], \tag{4.8}$$

where t_i^q and t_i^d denote the i^{th} term in query q and document d, respectively. For the network of the *rankprob* model, another similar term is concatenated with the above vector for the second document. The embedding function $\mathcal{E}$ transforms each term to a dense m-dimensional float vector as its representation, which is learned during the training phase. The weighting function $\mathcal{W}$ assigns a weight to each term in the vocabulary set, which is supposed to learn term global importance for the retrieval task. The compositionality function $\odot$ projects a set of n embedding and weighting pairs to an m-dimensional representation, independent from the value of n. The compositionality function is given by:

$$\odot_{i=1}^{n} (\mathcal{E}(t_i), \mathcal{W}(t_i)) = \sum_{i=1}^{n} \widehat{\mathcal{W}}(t_i) \cdot \mathcal{E}(t_i), \tag{4.9}$$

which is the weighted element-wise sum of the terms' embedding vectors. $\widehat{\mathcal{W}}$ is the normalized weight that is learned for each term, given as follows:

$$\widehat{\mathcal{W}}(t_i) = \frac{\exp(\mathcal{W}(t_i))}{\sum_{j=1}^{n} \exp(\mathcal{W}(t_j))}. \tag{4.10}$$

All combinations of different ranking architectures and different input representations presented in this section can be considered for developing ranking models.

4.2.4 Training Neural Rankers with Weak Supervision

In this section, we discuss the effectiveness of our neural rankers with different learning objectives (Section 4.2.2) and different input representations (Section 4.2.3), when they are trained with weakly supervised signals.

In the following, we first describe the train and evaluation data, metrics we report, and detailed experimental setup. Then we discuss the results.

Table 4.1: Collections statistics.

Collection	Genre	Queries	# docs	length
Robust04	news	301–450,601–700	528k	254
ClueWeb	webpages	1–200	50m	1,506

Collections

In our experiments, we used two standard TREC collections: The first collection (called *Robust04*[2]) consists of over 500k news articles from different news agencies, that is available in TREC Disks 4 and 5 (excluding Congressional Records). This collection, which was used in TREC Robust Track 2004, is considered as a homogeneous collection, because of the nature and the quality of documents. The second collection (called *ClueWeb*[3]) that we used is ClueWeb09 Category B, a large-scale web collection with over 50 million English documents, which is considered as a heterogeneous collection. This collection has been used in the TREC Web Track, for several years. In our experiments with this collection, we filtered out the spam documents using the Waterloo spam scorer[4] [57] with the default threshold 70%. The statistics of these collections are reported in Table 4.1.

Training Query Set

To train our neural ranking models, we used the unique queries (only the query string) appearing in the AOL query logs [215]. This query set contains web queries initiated by real users in the AOL search engine that were sampled from a three-month period from March 1, 2006 to May 31, 2006. We filtered out a large volume of navigational queries containing URL substrings ("http", "www.", ".com", ".net", ".org", ".edu"). We also removed all non-alphanumeric characters from the queries. We made sure that no queries from the training set appear in our evaluation sets. For each dataset, we took queries that have at least ten hits in the target corpus using the pseudo-labeler method. Applying all these processes, we ended up with 6.15 million queries for the Robust04 dataset and 6.87 million queries for the ClueWeb dataset. In our experiments, we randomly selected 80% of the training queries as training set and the remaining 20% of the queries were chosen as the validation set for hyper-parameter tuning. As the "pseudo-labeler" in our training data, we have used BM25 to score/rank documents in the collections given the queries in the training query set.

[2]*https://trec.nist.gov/data/robust/04.guidelines.html*
[3]*https://lemurproject.org/clueweb09/*
[4]*http://plg.uwaterloo.ca/~gvcormac/clueweb09spam/*

Evaluation Query Sets

We use the following query sets for evaluation that contain human-labeled judgments: a set of 250 queries (TREC topics 301–450 and 601–700) for the Robust04 collection that were previously used in TREC Robust Track 2004. A set of 200 queries (topics 1–200) were used for the experiments on the ClueWeb collection. These queries were used in the TREC Web Track 2009–2012. We only used the title of topics as queries.

Evaluation Metrics

To evaluate retrieval effectiveness, we report three standard evaluation metrics: mean average precision (MAP) of the top-ranked $1,000$ documents, precision of the top-20 retrieved documents (P@20), and normalized discounted cumulative gain (nDCG) [149] calculated for the top-20 retrieved documents (nDCG@20). Statistically significant differences of MAP, P@20, and nDCG@20 values are determined using the two-tailed paired t-test with $p\_value < 0.05$, with Bonferroni correction.

Experimental Setup

All models described in Section 4.2.2 are implemented using TensorFlow [98, 273]. In all experiments, the parameters of the network are optimized employing the Adam optimizer [161] and using the computed gradient of the loss to perform the back-propagation algorithm. All model hyper-parameters were tuned on the respective validation set using batched GP bandits with an expected improvement acquisition function [87]. For each model, the size of hidden layers and the number of hidden layers were selected from $[16, 32, 64, 128, 256, 512, 1024]$ and $[1, 2, 3, 4]$, respectively. The initial learning rate and the dropout parameter were selected from $[1 \times 10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-4}, 5 \times 10^{-5}, 1 \times 10^{-5}]$ and $[0.0, 0.1, 0.2, 0.5]$, respectively. For models with embedding vector representation, we considered embedding sizes of $[100, 300, 500, 1000]$. As the training data, we take the top-$1,000$ retrieved documents for each query from training query set Q, to prepare the training data. In total, we have $|Q| \times 1000$ ($\sim 6 \times 10^{10}$ examples in our data) point-wise example and $\sim |Q| \times 1000^2$ ($\sim 6 \times 10^{13}$ examples in our data) pair-wise examples. The batch size in our experiments was selected from $[128, 256, 512]$. At inference time, for each query, we take the top-$2,000$ retrieved documents using BM25 as candidate documents and re-rank them by the trained models. In our experiments, we use the Indri[5]

[5]*https://www.lemurproject.org/indri.php*

Table 4.2: Performance of the different models on different datasets. $^{\blacktriangle}$ or $^{\triangledown}$ indicates that the improvements or degradations with respect to BM25 are statistically significant, at the 0.05 level using the paired two-tailed t-test.

Method	**Robust04**			**ClueWeb**		
	MAP	*P@20*	*nDCG@20*	*MAP*	*P@20*	*nDCG@20*
BM25	0.2503	0.3569	0.4102	0.1021	0.2418	0.2070
Score + Dense	0.1961$^{\triangledown}$	0.2787$^{\triangledown}$	0.3260$^{\triangledown}$	0.0689$^{\triangledown}$	0.1518$^{\triangledown}$	0.1430$^{\triangledown}$
Score + Sparse	0.2141$^{\triangledown}$	0.3180$^{\triangledown}$	0.3604$^{\triangledown}$	0.0701$^{\triangledown}$	0.1889$^{\triangledown}$	0.1495$^{\triangledown}$
Score + Embed	0.2423$^{\triangledown}$	0.3501	0.3999	0.1002	0.2513	0.2130
Rank + Dense	0.1940$^{\triangledown}$	0.2830$^{\triangledown}$	0.3317$^{\triangledown}$	0.0622$^{\triangledown}$	0.1516$^{\triangledown}$	0.1383$^{\triangledown}$
Rank + Sparse	0.2213$^{\triangledown}$	0.3216$^{\triangledown}$	0.3628$^{\triangledown}$	0.0776$^{\triangledown}$	0.1989$^{\triangledown}$	0.1816$^{\triangledown}$
Rank + Embed	**0.2811**$^{\blacktriangle}$	**0.3773**$^{\blacktriangle}$	**0.4302**$^{\blacktriangle}$	**0.1306**$^{\blacktriangle}$	**0.2839**$^{\blacktriangle}$	**0.2216**$^{\blacktriangle}$
RankProb + Dense	0.2192$^{\triangledown}$	0.2966$^{\triangledown}$	0.3278$^{\triangledown}$	0.0702$^{\triangledown}$	0.1711$^{\triangledown}$	0.1506$^{\triangledown}$
RankProb + Sparse	0.2246$^{\triangledown}$	0.3250$^{\triangledown}$	0.3763$^{\triangledown}$	0.0894$^{\triangledown}$	0.2109$^{\triangledown}$	0.1916
RankProb + Embed	**0.2837**$^{\blacktriangle}$	**0.3802**$^{\blacktriangle}$	**0.4389**$^{\blacktriangle}$	**0.1387**$^{\blacktriangle}$	**0.2967**$^{\blacktriangle}$	**0.2330**$^{\blacktriangle}$

implementation of BM25 with the default parameters (i.e., $k_1 = 1.2$, $b = 0.75$, and $k_3 = 1000$).

Given the setup we explained above, we train and evaluate our neural rankers to address our second research question in this chapter:

> **RQ-2.1.2** *What setup in terms of input representation and learning objective is most suitable for a neural ranker when training on programmatically generated labeled data?*

We attempt to break down our experiments and analyses to different parts addressing several subquestions, and provide empirical answers along with the intuition and analysis behind each question:

How do the neural models with different training objectives and input representations compare?

Table 4.2 presents the performance of all model combinations. Interestingly, combinations of the *rank* model and the *rankprob* model with embedding vector representation outperform BM25 by significant margins in both collections. For instance, the *rankprob* model with embedding vector representation that shows the best performance among the other methods, surprisingly, improves BM25 by over 13% and 35% on the Robust04 and ClueWeb collections, respectively, in terms of MAP. Similar improvements can be observed for the other evaluation metrics.

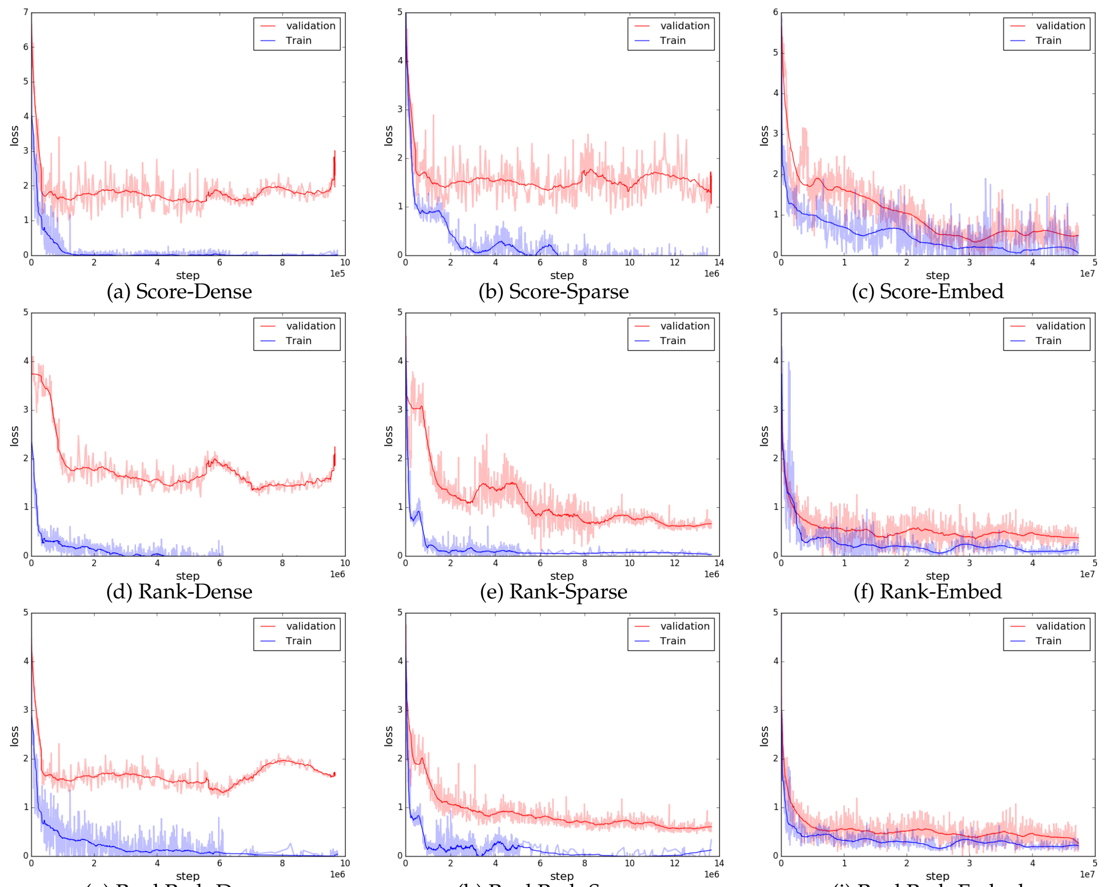
Regarding the modeling architecture, in the *rank* model and the *rankprob* model, compared to the *score* model, we define objective functions that target to learn ranking instead of scoring. This is particularly important in weak supervision, as the scores are imperfect values—using the ranking objective alleviates this issue by forcing the model to learn a preference function rather than reproduce absolute scores. In other words, using the ranking objective instead of learning to predict calibrated scores allows the *rank* model and the *rankprob* model to learn to distinguish between examples whose scores are close. This way, some small amount of noise, which is a common problem in weak supervision, would not perturb the ranking as easily [323].

Regarding the input representations, embedding vector representation leads to better performance compared to the other ones in all models. Using embedding vector representation not only provides the network with more information, but also lets the network to learn proper representation capturing the needed elements for the next layers with a better understanding of the interactions between query and documents. Providing the network with already engineered features would block it from going beyond the weak supervision signal and limit the ability of the models to learn latent features that are unattainable through feature engineering.

Note that although the *rankprob* model is more precise in terms of MAP, the *rank* model is much faster at inference time ($O(n)$ compared to $O(n^2)$), which is a desirable property in real-life applications.

Why do dense vector representation and sparse vector representation fail to replicate the performance of BM25?

Although neural networks are capable of approximating arbitrarily complex non-linear functions, we observe that the models with dense vector representation fails to replicate the BM25 performance, while they are given the same feature inputs as the BM25 components (e.g., TF, IDF, average document length, etc). To ensure that the training converge and there is no overfitting, we have looked into the training and validation loss values of different models during the training time. Figure 4.2 illustrates the loss curves for the training and validation sets per training step for different models. As shown, in models with dense vector representations, the training losses drop quickly to values close to zero while this is not the case for the validation losses, which is an indicator of over-fitting on the training data. Although we have tried different regularization techniques, like l_2-regularization and dropout with various parameters, there is less chance for generalization when the networks are fed with the fully featurized input. Note that over-fitting would lead to poor performance, especially in weak supervision scenarios as the network learns to



(a) Score-Dense (b) Score-Sparse (c) Score-Embed
(d) Rank-Dense (e) Rank-Sparse (f) Rank-Embed
(g) RankProb-Dense (h) RankProb-Sparse (i) RankProb-Embed

Figure 4.2: Training and validation loss curves for all combinations of different ranking architectures and input representations.

model imperfection of weak annotations. This phenomenon is also the case for models with the sparse vector representations, but with less impact. However, in models with the embedding vector representations, the networks do not overfit, which helps it to go beyond the weak supervision signals in the training data.

How are the models related?

To better understand the relationship between different neural models described above, we compare their performance across the query dimension following the approach in [200]. We assume that similar models should perform similarly for the same queries. Hence, we represent each model by a vector, called the performance vector, whose elements correspond to per query performance of the model, in terms of nDCG@20. The closer the performance vectors are, the more similar the models are in terms of the query by query performance. For the sake of visualization, we reduce the vectors dimension by projecting them to

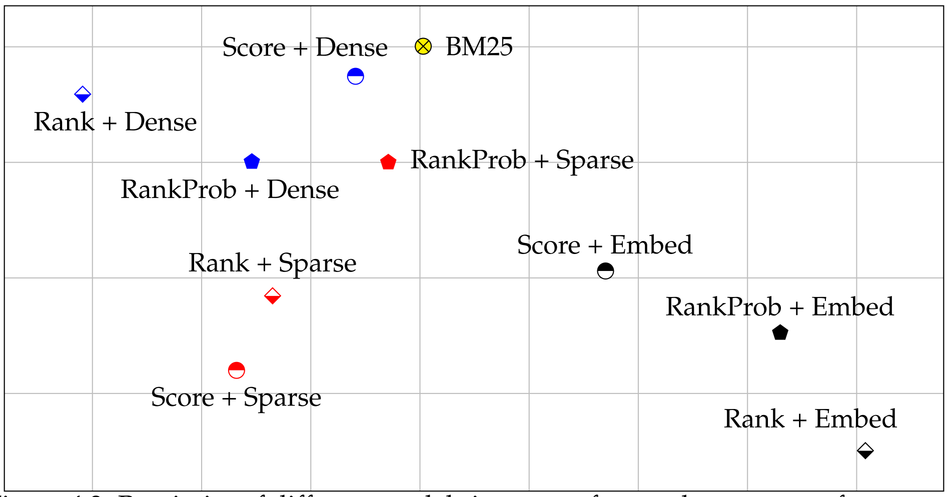


Figure 4.3: Proximity of different models in terms of query-by-query performance.

a two-dimensional space, using t-Distributed Stochastic Neighbor Embedding (t-SNE).[6]

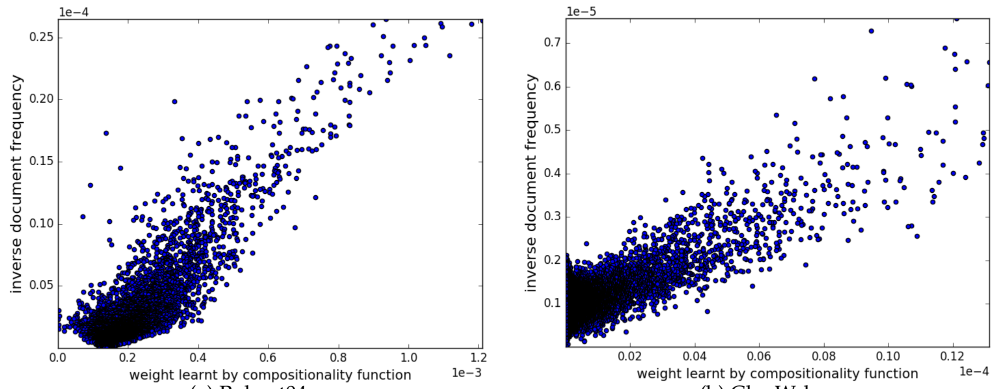
Figure 4.3 illustrates the proximity of different models according to their performances on the Robust04 collection. Based on this plot, models with similar input representations (same color) have quite close performance vectors, which means that they perform similarly for the same queries. This is not necessarily the case for models with similar architectures (same shape). This suggests that the amount and the way that we provide information to the networks are the key factors in the ranking performance.

We also observe that the *score* model with dense vector representations is the closest to BM25, which is expected. It is also interesting that models with embedding vector representation are placed far away from other models which shows they perform differently compared to the other input representations.

How meaningful are the compositionality weights learned in the embedding vector representation?

In this experiment, we focus on the best performing combination, i.e., the *rankprob* model with embedding vector representations. To analyze what the network learns, we look into the weights $\mathcal{W}$ (see Section 4.2.3) learned by the network. Note that the weighting function $\mathcal{W}$ learns a global weight for each vocabulary term. We notice that in both collections there is a strong linear correlation between the learned weights and the inverse document Figure 4.4 illustrates the scatter plots of the learned weight for each vocabulary term and its IDF, in both collections. This is an interesting observation as we do

[6] *https://lvdmaaten.github.io/tsne/*



(a) Robust04 (Pearson Correlation: 0.8243)

(b) ClueWeb (Pearson Correlation: 0.7014)

Figure 4.4: Strong linear correlation between weight learned by the compositionality function in the embedding vector representation and inverse document frequency.

Table 4.3: Performance of the *rankprob* model with variants of the embedding vector representation on different datasets. $^\blacktriangle$ indicates that the improvements over all other models are statistically significant, at the 0.05 level using the paired two-tailed t-test, with Bonferroni correction.

Embedding type	**Robust04**			**ClueWeb**		
	MAP	*P@20*	*nDCG@20*	*MAP*	*P@20*	*nDCG@20*
Pretrained (external) + Uniform weighting	0.1656	0.2543	0.3017	0.0612	0.1300	0.1401
Pretrained (external) + IDF weighting	0.1711	0.2755	0.3104	0.0712	0.1346	0.1469
Pretrained (external) + Weight learning	0.1880	0.2890	0.3413	0.0756	0.1344	0.1583
Pretrained (target) + Uniform weighting	0.1217	0.2009	0.2791	0.0679	0.1331	0.1587
Pretrained (target) + IDF weighting	0.1402	0.2230	0.2876	0.0779	0.1674	0.1540
Pretrained (target) + Weight learning	0.1477	0.2266	0.2804	0.0816	0.1729	0.1608
Learned + Uniform weighting	0.2612	0.3602	0.4180	0.0912	0.2216	0.1841
Learned + IDF weighting	0.2676	0.3619	0.4200	0.1032	0.2419	0.1922
Learned + Weight learning	**0.2837**$^\blacktriangle$	**0.3802**$^\blacktriangle$	**0.4389**$^\blacktriangle$	**0.1387**$^\blacktriangle$	**0.2967**$^\blacktriangle$	**0.2330**$^\blacktriangle$

not provide any global corpus information to the network in training and the network is able to infer such global information by only observing individual training samples.

How well do alternatives for the embedding and weighting functions in the embedding vector representation perform?

Considering the embedding vector representation as the input representation, we have examined different alternatives for the embedding function $\mathcal{E}$: (1) employing pre-trained word embeddings learned from an external corpus (we used Google News), (2) employing pre-trained word embeddings learned from the target corpus (using the skip-gram model [195]), and (3) learning embeddings during the training as explained in Section 4.2.3. Furthermore, for the compositionality function $\odot$, we tried different alternatives: (1) uniform weighting (simple averaging which is a common approach in compositionality function),

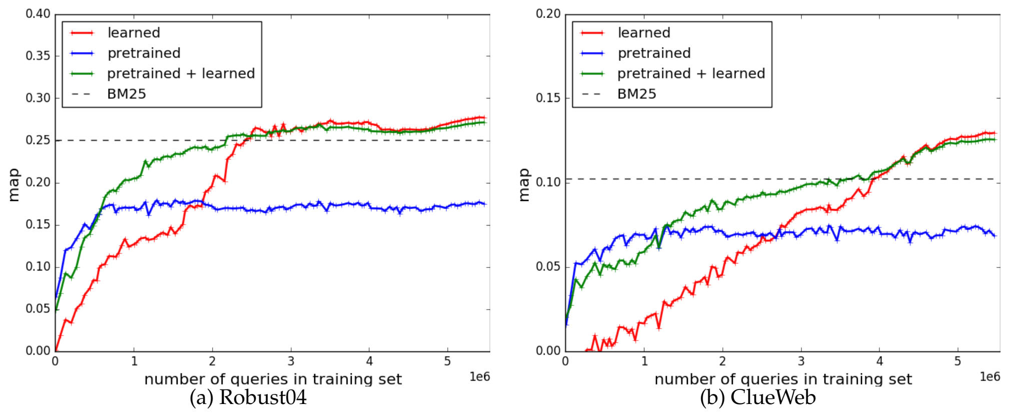


Figure 4.5: Performance of the *rankprob* model with learned embeddings, pre-trained embeddings, and learned embeddings with pre-trained embeddings as initialization, with respect to different amounts of training data.

(2) using IDF as fixed weights instead of learning the weighting function $\mathcal{W}$, and (3) learning weights during the training as described in Section 4.2.3.

Table 4.3 presents the performance of all these combinations on both collections. We note that learning both embeddings and weighting functions leads to the highest performance on both collections. These improvements are statistically significant. Regardless of the weighting approach, learning embeddings during training outperforms the models with fixed pre-trained embeddings. This supports the hypothesis that with the embedding vector representation the neural networks learn an embedding that is based on the interactions of query and documents that tends to be tuned better to the corresponding ranking task. Also, regardless of the embedding method, learning weights helps models to get better performance compared to the fixed weightings, with either IDF or uniform weights. Although weight learning can significantly affect performance, it has less impact than learning embeddings.

Note that in the models with pre-trained word embeddings, employing word embeddings trained on the target collection outperforms those trained on the external corpus in the ClueWeb collection; while this is not the case for the Robust04 collection. The reason could be related to the collection size, since the ClueWeb is approximately 100 times larger than the Robust04.

In addition to the aforementioned experiments, we have also tried initializing the embedding matrix with a pre-trained word embedding trained on the Google News corpus, instead of random initialization. Figure 4.5 presents the learning curve of the models. According to this figure, the model initialized by a pre-trained embedding performs better than random initialization when a limited amount of training data is available. When enough training data is fed to the network, initializing with pre-trained embedding and random values converge to the same performance. An interesting observation here is that in

Table 4.4: Performance of the linear RankSVM with different features.

Method	Robust04			ClueWeb		
	MAP	*P@20*	*nDCG@20*	*MAP*	*P@20*	*nDCG@20*
RankSVM + Dense	0.1983	0.2841	0.3375	0.0761	0.1840	0.1637
RankSVM + Sparse	0.2307	0.3260	0.3794	0.0862	0.2170	0.1939
RankSVM + (Pretrained (external) + IDF weighting)	0.1539	0.2121	0.1852	0.0633	0.1572	0.1494
Score (one layer with no nonlinearity) + Embed	0.2103	0.3986	0.3160	0.0645	0.1421	0.1322

both collections, these two initializations converge when the models exceed the performance of the weak supervision source, which is BM25 in our experiments. This suggests that convergence occurs when accurate representations are learned by the networks, regardless of the initialization.

Are deep neural networks a good choice for learning to rank with weak supervision?

To see if there is a real benefit from using a non-linear neural network in different settings, we examined RankSVM [150] as a strong-performing pair-wise learning to rank method with a linear kernel that is fed with different inputs: dense vector representations, sparse vector representations, and embedding vector representations. Considering that off-the-shelf RankSVM is not able to learn embedding representations during training, we use a pre-trained embedding matrix trained on Google News and fixed IDF weights.

The results are reported in Table 4.4. As BM25 is not a linear function, RankSVM with the linear kernel is not able to completely approximate it. However, surprisingly, for both dense vector representations and sparse vector representations, RankSVM works as well as neural networks (see Table 4.2). Also, compared to the corresponding experiment in Table 4.3, the performance of the neural network with an external pre-trained embedding and IDF weighting is not considerably better than RankSVM. This shows that having non-linearity in neural networks does not help that much when we do not have representation learning as part of the model. Note that all of these results are still lower than BM25, which shows that they are not good at learning from weak supervision signals for ranking.

We have also examined the *score* model with a network with a single linear hidden layer, with the embedding vector representation, which is equivalent to a linear regression model with the ability of representation learning. Comparing the results of this experiment with Score-Embed in Table 4.2, we can see that with a single-linear network we are not able to achieve a performance that is as good as a deep neural network with non-linearity. This shows that the most important superiority of deep neural networks over other machine learning methods is their ability to learn an effective representation and take all the inter-

Table 4.5: Performance of the *rankprob* model with embedding vector representation in fully supervised setting, weak supervised setting, and weak supervised plus supervision as fine tuning. ▲ indicates that the improvements over all other models are statistically significant, at the 0.05 level using the paired two-tailed t-test, with Bonferroni correction.

Method	**Robust04**			**ClueWeb**		
	MAP	*P@20*	*nDCG@20*	*MAP*	*P@20*	*nDCG@20*
Weakly supervised	0.2837	0.3802	0.4389	0.1387	0.2967	0.2330
Fully supervised	0.1790	0.2863	0.3402	0.0680	0.1425	0.1652
Weakly supervised + Fully supervised	**0.2912**$^\blacktriangle$	**0.4126**$^\blacktriangle$	**0.4509**$^\blacktriangle$	**0.1520**$^\blacktriangle$	**0.3077**$^\blacktriangle$	**0.2461**$^\blacktriangle$

actions between query and document(s) into consideration for approximating an effective ranking/scoring function.

How useful is learning with weak supervision for supervised ranking?

In this set of experiments, we investigate whether employing weak supervision as a pre-training step helps to improve the performance of supervised ranking, when a small amount of training data is available. Table 4.5 shows the performance of the *rankprob* model with the embedding vector representation in three situations: (1) when it is only trained on weakly supervised data (similar to the previous experiments), (2) when it is only trained on supervised data, i.e., relevance judgments, and (3) when the parameters of the network are pre-trained using the weakly supervised data and then fine-tuned using relevance judgments. In all supervised scenarios, we performed 5-fold cross-validation over the queries of each collection and in each step, we used the TREC relevance judgments of the training set as a supervised signal. For each query with m relevant documents, we also randomly sampled m non-relevant documents as negative samples. Binary labels are used in the experiments: 1 for relevant documents and 0 for non-relevant ones.

The results in Table 4.5 suggest that pre-training the network with a weak supervision signal, significantly improves the performance of supervised ranking. The reason for the poor performance of the supervised model compared to the conventional learning to rank models is that the number of parameters is much larger, hence it needs much more data for training.

In situations when little supervised data is available, it is especially helpful to use unsupervised pre-training which acts as a network pre-conditioning that puts the parameter values in the appropriate range that renders the optimization process more effective for further supervised training [97].

With this experiment, we indicate that the idea of learning from weak supervision signals for neural ranking models, which is presented in this section, not only enables us to learn neural ranking models when no supervised signal is available, but also has substantial positive effects on the supervised ranking models with limited amount of training data.

In the next section, we study how learning from weak/noisy label can help preserve privacy in machine learning, like in situations where additional noise is intentionally added to the training data to guarantee differential privacy.

4.3 Weakly Supervised Learning for Preserving Privacy

Deep neural networks perform better as the training dataset grows bigger and becomes diverse and representative [269] of all possible cases. In many applications, the datasets that are big and diverse may contain sensitive information from users, for instance, medical histories of patients in a clinical trial, or search logs from users of a search engine. It has been shown that a trained model may inadvertently and implicitly store some of its training data and we can retrieve some of the information about samples in the training data [256], either directly by analyzing internal model parameters or indirectly by repeatedly querying the model as a black-box to gather data and do analysis on those data [102].

We need to design and use learning algorithms that protect the privacy of users, for instance, by guaranteeing that the output model generalizes away from the specifics of any individual user. Recently, Papernot et al. [212] proposed Private Aggregation of Teacher Ensembles (PATE), a generally applicable approach to providing strong privacy guarantees for training data. PATE uses a noisy aggregation of the signal that comes from multiple models trained with disjoint datasets, to train a new model that guarantees a certain level of differential privacy.

Almost all deferentially private algorithms add noise to introduce ambiguity. Hence, the training signals become less perfect and employing noise-robust models can support injecting noise, yielding strong privacy guarantees, while having a limited impact on accuracy.

Search and retrieval is one of the applications that needs special attention on preserving the privacy of users' data and many recent advances rely on sensitive and private data such as large-scale query logs, users' search history, and location information [312]. In this section, we focus on the task of ranking and assessing the relevance. Here we seek an answer to the third research question of this chapter:

> **RQ-2.1.3** *How can learning from weak supervision signals help to preserve privacy while training neural networks on sensitive data?*

We present the results of a set of experiments that examine the performance of one of the neural ranking architectures proposed in Section 4.2 when it is employed in PATE, the privacy-preserving framework proposed by Papernot et al. [212], where the neural ranker is supposed to learn from signals with added noise. Since PATE is based on the knowledge distillation framework [137], we first train a neural ranker in a mimic-learning setup where a student ranker is trained on the signals from a teacher ranker that is trained using labeled data. Then we use the full privacy preserving pipeline of PATE to train our neural ranker. It is noteworthy that here, we mainly concern about the performance of a neural ranker, when it is employed in the PATE's setup and will not re-discuss the differential privacy of PATE, as this side of the discussion is presented thoroughly in the original paper [212].

4.3.1 Mimic Learning to Rank

Using machine learning-based approaches, sharing the trained model instead of the original data has turned out to be an option for transferring knowledge [1, 212, 256]. The idea of *mimic learning* is to use a model that is trained based on the signals from the original training data to annotate a large set of unlabeled data and use these labels as training signals for training a new model. It has been shown, for many tasks in computer vision and natural language processing, that we can transfer knowledge this way and the newly trained models perform as well as the model trained on the original training data [17, 33, 137, 235].

We follow the knowledge distillation approach [137] for training a neural ranker, where we have a teacher network that is trained using labeled data, and a student network that is trained using the signals from the teacher network on a set of unlabeled data. We have two sets of experiments, in the first one, we train the teacher model with full supervision, i.e., on the set of queries with judgments, using 5-fold cross-validation. In the second set of experiments, the set of queries with judgments is only used for evaluation and we train the teacher model using the weak supervision setup, i.e., pseudo labels as it is explained in Section 4.2.1. As the test collection, we use Robust04, which has been introduced in Section 4.2.4. In all experiments, we use a separate set of 3 million queries from the AOL query log, preprocessed as explained in Section 4.2.4.

In all experiments, as the neural rankers, we use *rank* model,[7] which has been described in Section 4.2.2, with embedding vector representation as the input

[7] Although in Section 4.2.4 we showed that the best performing model is the *rankprob* model, its improvement over the *rank* model was not statistically significant. Compared to the *rankprob* model, the *rank* model is much more efficient at inference time, as it operates in a point-wise setup, unlike the *rankprob* model which runs in a pair-wise mode. We decided to adapt the *rank* model for all experiments in this section.

Table 4.6: Teacher and student neural networks configurations.

Parameter	**Teacher**	**Student**
Number of hidden layers	3	3
Size of hidden layers	512	128
Initial learning rate	1E-3	1E-3
Dropout	0.2	0.1
Embedding size	500	300
Batch size	512	512

Table 4.7: Performance of teacher and student models with different training strategies.

Training strategy	**model**	**MAP**	**P@20**	**nDCG@20**
Full supervision	*Teacher*	0.1814	0.2888	0.3419
	Student	0.2256	0.3111	0.3891
Weak supervision	*Teacher*	0.2716	0.3664	0.4109
	Student	0.2701	0.3562	0.4145

representation, which has been explained in Section 4.2.3. The configuration of teacher and student networks is presented in Table 4.6.

Results obtained from these experiments are summarized in Table 4.7. As the results suggest, using weak supervision to train the teacher model, the student model performs as good as the teacher model. In case of training the teacher with full supervision (labeled data from Robust04), as the original training data is small, the performance of the teacher model is rather low, which is mostly due to the fact that the big teacher model overfits on the train data and is not able to generalize well. However, due to the regularization effect of the knowledge distillation process, the student model, which is trained on the predictions by the teacher model significantly outperforms the teacher model [137, 235].

4.3.2 Privacy Preserving Neural Ranker

In Section 4.3.1, we examined the idea of mimic learning to train a neural ranker regardless of the privacy concerns. It has been shown that there is a risk of privacy problems, both where the adversary is just able to query the model, and where the model parameters are exposed to the adversaries inspection. For instance, Fredrikson et al. [102] show that only by observing the prediction of the machine learning models they can already approximately reconstruct part of the training data (model-inversion attack). Shokri et al. [257] also demonstrate that it is possible to infer whether a specific training point is included in the model's training data by observing only the predictions of the

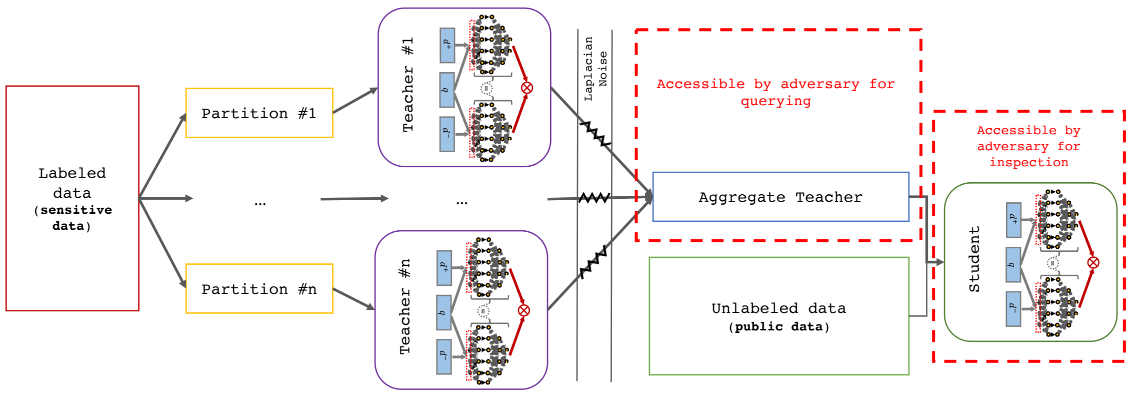

Labeled data (sensitive data)
Partition #1
...
Partition #n
Teacher #1
...
Teacher #n
Laplacian Noise
Accessible by adversary for querying
Aggregate Teacher
Unlabeled data (public data)
Accessible by adversary for inspection
Student

Figure 4.6: Privacy preserving annotator/model sharing, proposed by Papernot et al. [212].

model (membership inference attack).

In this section, we adapt the Private Aggregation of Teacher Ensembles (PATE) [212] to train a privacy-preserving neural ranking model. PATE is based on the student-teacher framework [137], where there are multiple teacher models trained on disjoint subsets of the data and a student model that learns to predict an output that is chosen by noisy voting among all of the teachers. The student model cannot directly access an individual teacher or the underlying data or parameters. They show that PATE improves privacy/utility trade-offs by achieving high accuracy while guaranteeing a certain level of differential privacy.

The general schema of the PATE is illustrated in Figure 4.6. First, the sensitive training data is divided into n partitions. Then, on each partition, an independent neural network model is trained as a teacher. Once all the teachers are trained, an aggregation step is done using majority voting to generate a single global prediction. Laplacian noise is injected into the output of the prediction of each teacher before aggregation. The introduction of this noise is what protects privacy because it obfuscates the vulnerable cases, where teachers disagree.

The aggregated teacher can be considered as a deferentially private API to which we can submit the input and it then returns the privacy-preserving label. There are some circumstances where due to efficiency reasons the model needs to be deployed on the user device [1]. To be able to generate a shareable model where the privacy of the training data is preserved, Papernot et al. [212] train an additional model called the student model. The student model has access to unlabeled public data during training. The unlabeled public data is annotated using the aggregated teacher to transfer knowledge from teachers to student model in a privacy-preserving fashion. This way, if the adversary tries to recover the training data by inspecting the parameters of the student model, in the worst case, the public training instances with privacy-preserving labels from the aggregated teacher are going to be revealed. The privacy guarantee of

Table 4.8: Performance of the teachers (average) and student models with noisy and non-noisy aggregation.

Model	**MAP**	**P@20**	**nDCG@20**
Teachers (avg)	0.2566	0.3300	0.3836
Non-noisy aggregated teacher	0.2380	0.3055	0.3702
Student (non-noisy aggregation)	0.2337	0.3192	0.3717
Noisy aggregated teacher	0.2110	0.2868	0.3407
Student (noisy aggregation)	0.2255	0.2984	0.3559

this approach is formally proved using the differential privacy framework.

As there is no publicly available large-scale data that we can use in our experiments as the initial sensitive labeled data, we use pseudo labels as it is explained in 4.2.1.[8] In our experiments, we split the training data into three partitions, each contains one million queries annotated by the BM25 method. We train three identical teacher models. Then, we use the aggregated noisy predictions from these teachers to train the student network using the knowledge distillation approach. Configurations of teacher and student networks are similar to the previous experiments, as they are presented in Table 4.6.

We evaluate the performance in two situations: In the first one, the privacy parameter, which determines the amount of noise, is set to zero, and in the second one, the noise parameter is set to 0.05, which guarantees a low privacy risk [212]. We report the average performance of the teachers before noise, the performance of noisy and non-noisy aggregated teacher, and the performance of the student networks in two situations. The results of these experiments are reported in Table 4.8.

Results in Table 4.8 suggest that by using the noisy aggregation of multiple teachers as the supervision signal, we can train a neural ranker with acceptable performance. Compared to the single teacher setup in Section 4.3.1, the performance of the student network is not as good as the average performance of teachers, although the student network performs better than the teacher in the noisy aggregation setup. This is more or less the case for a student together with a non-noisy aggregated teacher. We believe drops in the performance on the student networks compared to the results in Section 4.3.1 are not just due to partitioning, noise, and aggregation. This is also the effect of the change in the amount of training data for the teachers in our experiments. We expect that in the case of having enough training data in each partition for each teacher, their prediction will be more determined and we will have less disagreement in the

[8]Partitioning the fully supervised training data in our problem leads to very small training sets which are not big enough to train teacher networks.

aggregation phase and consequently, we will get better signals for training the student model.

4.4 Related Work

In this section, we briefly review the neural ranking models in terms of their general architectures and discuss how the neural models proposed in this chapter are related to the previous works.

Recently, several attempts have been made to study deep neural networks in IR applications, which can be generally partitioned into two categories [119, 209, 331]. The first category includes approaches that use the results of trained (deep) neural networks in order to improve the performance in IR applications. Among these, distributed word representations or embeddings [195, 218] have attracted a lot of attention. Word embedding vectors have been applied to term re-weighting in IR models [332], query expansion [93, 320], query classification [179, 321], etc. The main shortcoming of most of the approaches in this category is that the objective of the trained neural network differs from the objective of these tasks. For instance, the word embedding vectors proposed in [195, 218] are trained based on term proximity in a large corpus, which is different from the objective in most IR tasks. Zamani and Croft [322] recently proposed relevance-based word embedding models for learning word representations based on the objectives that matter for IR applications.

The second category, which the models proposed in this chapter belong to, consists of approach that design and train a (deep) neural network for a specific task, e.g., question answering [52, 313], click models [27]. A number of the approaches in this category have been proposed for ranking documents in response to a given query. These approaches can generally be divided into two groups: *late combination models* and *early combination models* (or representation-focused and interaction-focused models according to [118]). The late combination models, following the idea of Siamese networks [32], independently learn a representation for each query and candidate document and then calculate the similarity between the two estimated representations via a similarity function. For example, Huang et al. [144] proposed DSSM, which is a feed forward neural network with a word hashing phase as the first layer to predict the click probability given a query string and a document title. The DSSM model was further improved by incorporating convolutional neural networks [253].

In contrast, early combination models are designed based on interactions between the query and the candidate document as the input of a network. For instance, DeepMatch [182] maps each text to a sequence of terms and trains a feed-forward network for computing the matching score. The deep relevance matching model for ad-hoc retrieval [118] is another example of an early combination model that feeds a neural network with histogram-based features representing interactions between the query and document. Early combining enables the model to have an opportunity to capture various interactions between query and document(s), while with the late combination approach, the model has only the chance of encoding query and documents independently (at least at the early layers). Recently, Mitra et al. [200] proposed to simultaneously learn local and distributional representations, which are early and late combination models respectively, to capture both exact term matching and semantic term matching.

Until now, all the proposed neural models for ranking are trained on either explicit relevance judgements or clickthrough logs. However, a massive amount of such training data is not always available.

In this chapter, we propose to train neural ranking models using weak supervision, which is the most natural way to reuse existing supervised learning models where imperfect labels are treated as ground truth. The basic assumption is that we can cheaply obtain labels (that are of lower quality than human-provided labels) by expressing prior knowledge we have about the task at hand by specifying a set of heuristics, adapting existing ground truth data for a different but related task (this is often referred to a distant supervision[9]), extracting a supervision signal from external knowledge-bases or ontologies, crowd-sourcing partial annotations that are cheaper to get, etc. Weak supervision is a natural way to benefit from unsupervised data and it has been applied in NLP for various tasks including relation extraction [24, 123], knowledge-base completion [141], sentiment analysis [251], etc. There are also similar attempts in IR for automatically constructing test collections [10] and learning to rank using labeled features, i.e., features that an expert believes to be correlated with relevance [92]. In this chapter, we make use of traditional IR models as the weak supervision signal to generate a large amount of training data and train effective neural ranking models that outperform the baseline methods by a significant margin.

To circumvent the lack of human-labeled training examples, unsupervised learning methods aim to model the underlying data distribution, thus learning powerful feature representations of the input data, which can be helpful for building more accurate discriminative models especially when little or even no supervised data is available. A large group of unsupervised neural models seeks to exploit the implicit internal structure of the input data, which in turn requires customized formulation of the training objective (loss function), targeted network architectures and often non-trivial training setups.

[9]We do not distinguish between weak and distant supervision as the difference is subtle and both terms are often used interchangeably in the literature.

A lot of research has been done on the general problem of preserving the privacy of sensitive data in IR applications, where the question is how should we design effective IR systems without damaging users' privacy? One of the solutions so far is to anonymize the data and try to hide the identity of users [38, 331]. As an example, Zhang et al. [331] use a differential privacy approach for query log anonymization. However, there is no guarantee that the anonymized data will be as effective as the original data.

Modeling privacy in machine learning is a challenging problem and there has been much research in this area. Preserving the privacy of deep learning models is even more challenging, as there are more parameters to be safeguarded [220]. Some work has studied the vulnerability of deep neural network as a service, where the interaction with the model is only via an input-output black box [102, 257, 280]. Others have proposed approaches to protect privacy against an adversary with full knowledge of the training mechanism and access to the model's parameters. For instance, Abadi et al. [1] propose a privacy preserving stochastic gradient descent algorithm offering a trade-off between utility and privacy. More recently, Papernot et al. [212] propose a semi-supervised method for transferring the knowledge for deep learning from private training data. They propose a setup for learning privacy-preserving student models by transferring knowledge from an ensemble of teachers trained on disjoint subsets of the data for which privacy guarantees are provided. In this chapter, we followed their approach to train a privacy preserving neural ranker and showed that the ideas that we proposed in this chapter on training neural rankers with weak supervenes helps improve the performance when noise is intentionally added to the supervision signal.

4.5 Conclusion

In this chapter, we focused on addressing **RQ-2.1**: *"How can we train neural networks using programmatically generated pseudo-labels as a weak supervision signal, in a way that they exhibit superior generalization capabilities?"*. We proposed to use unsupervised methods in order to programmatically generate large amounts of training data [226], as weakly annotated data, to train effective neural ranking models. We focus on the task of assessing the relevance, i.e., ranking of documents given a query where no large-scale training set is publicly available. We examined various neural ranking models with different ranking architectures and objectives, and different input representations.

To investigate **RQ-2.1.1**, we used over six million queries to train our models and evaluated them on Robust04 and ClueWeb 09-Category B collections, in an ad-hoc retrieval setting. The experiments showed that our best performing model significantly outperforms the BM25 model (our weak supervision signal) by over 13% and 35% MAP improvements on the Robust04 and ClueWeb collections, respectively. We also demonstrated that in the case of having a small amount of training data, we can improve the performance of supervised learning by pre-training the network on weakly supervised data.

To address **RQ-2.1.2**, we showed that based on our results, there are three key ingredients in neural ranking models that lead to good performance with weak supervision: The first is the proper input representation. Providing the network with raw data and letting the network learn the features that matter, gives the network a chance of learning how to ignore imperfection of training data. The second ingredient is to target the right goal and define a proper objective function. In the case of having weakly annotated training data, by targeting some explicit labels from the data, we may end up with a model that learned to express the data very well, but is incapable of going beyond it. This is especially the case with deep neural networks where there are many parameters and it is easy to learn a model that overfits the data. The third ingredient is providing the network with a considerable amount of diverse training examples. As an example, during the experiments we noticed that using the embedding vector representation, the network needs a lot of examples to learn embeddings that are more effective for retrieval compared to pre-trained embeddings. Thanks to weak supervision, we can generate as much training data as we need with almost no cost.

To address **RQ-2.1.3**, we also study how learning from weak signals can benefit preserving privacy when some noise is intentionally added to the training signal. We employed Private Aggregation of Teacher Ensembles (PATE) [212] to train a privacy-preserving neural ranking model, in which we train several neural rankers on disjoint subsets of the training data and use the noisy aggregated signals from these models on an unlabeled set to train a neural ranker that in a setup that guarantees a certain level of differential privacy. These experiments lay the groundwork for the idea of sharing a privacy-preserving model instead of sensitive data in IR applications. This suggests researchers from industry are how able to share the knowledge learned from actual users' data with the academic community, without privacy risks, which leads to a better collaboration of all researchers in the field.

In this chapter, we mainly focused on the ranking task and explored architectural ideas that can implicitly help to have neural networks that are less sensitive to the label noise. In the next chapter, we propose more systematic approaches that are task and architecture independent and can learn to estimate the quality of the labels and explicitly control the learning process with respect to the estimated qualities.

5 Learning from Samples of Variable Quality

Training labels are expensive to obtain and may be of varying quality, as some may be from trusted expert labelers, while others might be from heuristics or other sources of weak supervision. This creates a fundamental quality-versus-quantity trade-off in the learning process. Do we learn from the small amount of high-quality data or the potentially large amount of weakly-labeled data? We argue that if the learner could somehow know and take the label-quality into account, we could get the best of both worlds.

5.1 Introduction

The success of deep neural networks to date depends strongly on the availability of labeled data [269]. The more neural networks become deep and complex, the more it is crucial for them to be trained on massive amounts of training data. However, in many applications, labeled data is costly to obtain and task-specific training data is now a critical bottleneck.

Usually, it is possible to obtain small quantities of high-quality labeled data and large quantities of unlabeled data. The problem of how to best integrate these two different sources of information during training is an active pursuit

This chapter is based on [79, 82, 83].

in the field of semi-supervised learning [40]. However, for a large class of tasks it is also easy to define one or more so-called "weak annotators," additional (albeit noisy) sources of *weak supervision* based on heuristics or "weaker," biased classifiers trained on, e.g., non-expert crowd-sourced data or data from related domains. While easy and cheap to generate, it is not immediately clear if and how these additional weakly-labeled data can be used to train a stronger classifier for the task we care about.

All labels are equal, but some labels are more equal than others.[1] This holds generally since in almost all practical applications, machine learning systems have to deal with data *samples of variable quality*. For example, in a large dataset of images, only a small fraction of samples may be labeled by experts and the rest may be crowd-sourced using, e.g., Amazon Mechanical Turk [293]. In addition, in some applications, labels are intentionally perturbed due to privacy issues [71, 213, 296], as we discussed in the Chapter 4.

Formally speaking, in our setup, we assume that we are given a large set of unlabeled data samples, a heuristic labeling function called the *weak annotator*, and a small set of high-quality samples labeled by experts, called the *strong dataset*, consisting of tuples of training samples x_j and their true labels y_j, i.e., $\mathcal{D}_s = \{(x_j, y_j)\}_{j=1}^{N}$. We consider the latter to be observations from the true target function that we are trying to learn. We use the weak annotator to generate labels for the unlabeled samples. Generated labels are noisy due to the limited accuracy of the weak annotator. This gives us a *weak dataset* consisting of tuples of training samples x_i and their weak labels $\tilde{y}_i$, i.e., $\mathcal{D}_w = \{(x_i, \tilde{y}_i)\}_{i=1}^{M}$. Note that we can generate a large amount of weak training data $\mathcal{D}_w$ at almost no cost using the weak annotator. In contrast, we only have a limited amount of observations from the true function, i.e., $|\mathcal{D}_s| \ll |\mathcal{D}_w|$.

The simplest approach in this setup is to expand the strong training set, $\mathcal{D}_s$, by including the weakly-supervised samples, $\mathcal{D}_w$, which comes down to considering all samples to be equally important. Alternatively, one may pretrain on the weak data and then fine-tune on observations from the true function or distribution. We showed in Chapter 4 that a small amount of expert-labeled data can be augmented in such a way by a large set of raw data, with labels coming from a heuristic function, to train a more accurate ranking model. The downside is that such approaches are oblivious to the amount or source of noise in the labels. Simply speaking, they do not consider the cause of noise in the labels and only focus on the effect.

In this chapter, we focus on this issue and try to address the following research question:

[1]Inspired by George Orwell quote, Animal Farm, 1945.

> **RQ-2.2** *Given a large set of weakly annotated samples and a small set of samples with high-quality labels, how can we best leverage the capacity of information in these sets to train a neural network?*

We argue that treating weakly-labeled samples uniformly (i.e., each weak sample contributes equally to the final classifier) ignores potentially valuable information of the label quality. Instead, we propose two different approaches that directly model the inaccuracies introduced by the weak annotator and estimate a "confidence" or "fidelity" score for each weak labeled sample, which can then be used to modulate the training process and control the extent to which we learn from these samples given their estimated confidence or fidelity scores.

5.1.1 Detailed Research Questions

We break down our main research question in this chapter into two concrete research questions:

> **RQ-2.2.1** *When learning from samples of variable quality, can we meta learn an adjustment for the magnitude of the parameter updates in back-propagation based on the merit of labels?*
>
> **RQ-2.2.2** *When learning from samples of variable quality, can we reannotate these samples and provide (hopefully) better labels, associated with a fidelity score to regulate the learning rate?*

In the following sections, we will address these research questions, and support our ideas with experiments and analyses on different tasks.

5.2 Learning to Learn from Weak Supervision, by Full Supervision

Using weak or noisy supervision is a straightforward approach to increase the size of the training data. This is usually done by pre-training the network on a large set of weakly labeled data and fine tuning it with strong labels [84, 250]. However, these two independent stages do not leverage the full capacity of information from the small set of strong labels, as it can be useful for learning how to learn from the weak labels. In particular, in the pre-training stage, we have to learn from labels of variable quality without any control over how these labels contribute to the learning process.

In this section, we address the first research question of this chapter:

RQ-2.2.1 *When learning from samples of variable quality, can we meta learn an adjustment for the magnitude of the parameter updates in back-propagation based on the merit of labels?*

We introduce a semi-supervised method that leverages a small amount of data with strong labels to improve the learning from a large amount of data with weak labels. This model, in fact, offers learning from *Controlled Weak Supervision* and we refer to it by *CWS* in the rest of this chapter. CWS has three main components: (1) A weak annotator, which can be a heuristic model, a weak or biased classifier, or even a human via crowd-sourcing and it is employed to annotate a massive amount of unlabeled data, (2) a target network, which uses a large set of weakly annotated samples by the weak annotator to learn the main task, (3) and a confidence network, which is trained on a small set with strong labels to estimate confidence scores for samples annotated by the weak annotator.

The confidence scores estimated by the confidence network define the magnitude of the weight updates applied to the target network during training. This way, the confidence network helps the target network to avoid the mistakes of its teacher, i.e., weak annotator, by down-weighting the weight updates from weak labels that do not look reliable according to confidence network. CWS, in fact, employs the teacher-student paradigm in which the target network (student) and the confidence network (teacher) are trained jointly in a multi-task fashion and they share parameters of the representation learning layer to share their understanding of the data.

From a meta-learning perspective [6, 100, 227], the goal of the confidence network —as the meta-learner— trained jointly with the target network —as the learner— is to calibrate the learning rate of the target network for each sample in the batch. I.e., the weights $\boldsymbol{w}$ of the target network f_w at step $t+1$ are updated as follows:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \frac{\eta_t}{b}\sum_{i=1}^{b} c_\theta(x_i, \tilde{y}_i)\nabla\mathcal{L}(f_{\boldsymbol{w}_t}(x_i), \tilde{y}_i), \tag{5.1}$$

where η_t is the global learning rate, $\mathcal{L}(\cdot)$ is the loss of predicting $\hat{y} = f_w(x_i)$ for an input x_i when $\tilde{y}$ is the weak label; $c_\theta(\cdot)$ is a scoring function learned by the confidence network taking input instance x_i and its noisy label $\tilde{y}_i$. Thus, we can effectively control the contribution to the parameter updates for the target network from weakly labeled samples based on how reliable their labels are according to the confidence network (learned on a small supervised data).

Our setup requires running a weak annotator to label a large amount of unlabeled data, which is done at pre-processing time. For many tasks, it is possible to use a simple heuristic, a rule-based function, or implicit human feedback to generate weak labels. This set is then used to train the target network. In contrast, a small expert-labeled set is used to train the confidence network, which estimates how good the weak annotations are, i.e., controls the effect of weak labels on updating the parameters of the target network.

CWS allows learning different types of neural architectures and various tasks, where a meaningful weak annotator is available. Later in this chapter we study the performance of CWS by focusing on two applications: sentiment classification and document ranking.

5.2.1 Learning from Controlled Weak Supervision

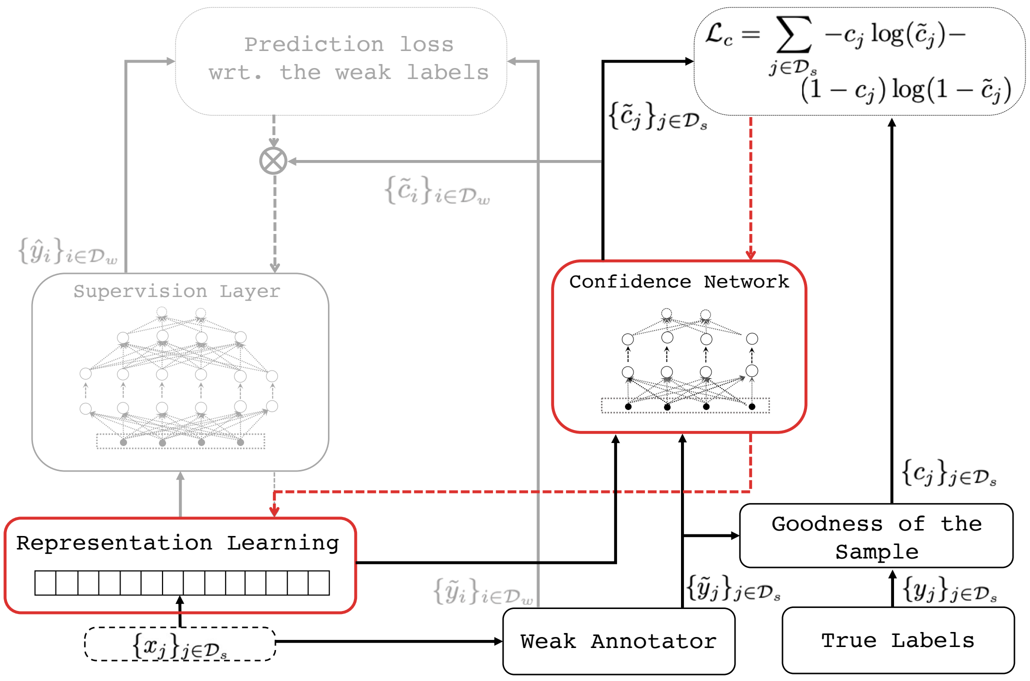
In the following, we describe our recipe for training a multi-task neural network that jointly learns the confidence score of weakly labeled samples and the main task using controlled supervised signals. The high-level representation of the model is shown in Figure 5.1.

Our model comprises a weak annotator and two neural networks: the confidence network and the target network. Formally speaking, the goal of the weak annotator is to *provide weak labels* $\tilde{y}_i$ for all the instances $x_i \in \mathcal{D}_w \cup \mathcal{D}_s$. We have the assumption that $\tilde{y}_i$ provided by the weak annotator are imperfect estimates of strong labels y_i, where y_i are available for the set $\mathcal{D}_s$, but not for the set $\mathcal{D}_w$.

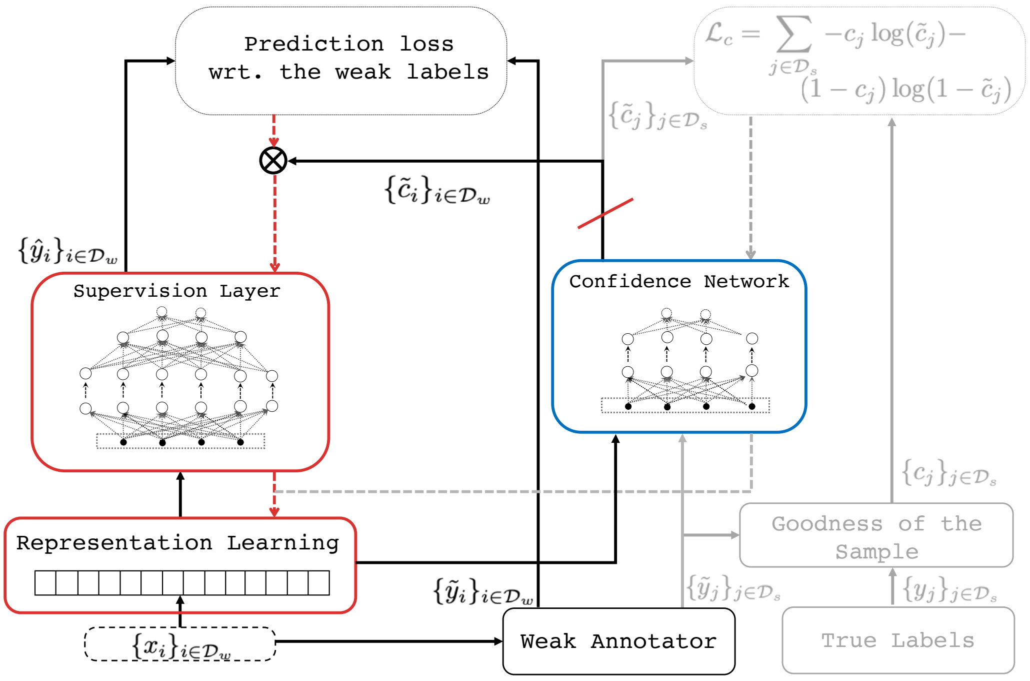
The goal of the confidence network is to *estimate the confidence score* $\tilde{c}_j$ of training samples. It is learned on samples from the training set $\mathcal{D}_s$, i.e a set of input x_j and its strong label y_j as well its weak label, $\tilde{y}_j$, that is annotated by the weak annotator. The score $\tilde{c}_j$ is then used to control the effect of weakly labeled samples on updating the parameters of the target network in the backward pass of backpropagation.

The target network is in charge of *handling the main task* we want to learn, or in other words, approximating the underlying function that predicts the correct labels. Given a data instance, x_i and its weak label $\tilde{y}_i$ from the training set $\mathcal{D}_w$, the target network aims to predict the label $\hat{y}_i$. The target network parameter updates are based on noisy labels assigned by the weak annotator, but the magnitude of the gradient update is based on the output of the confidence network.

Both networks are trained in a multi-task fashion alternating between the *full supervision* and the *weak supervision* mode. In the *full supervision* mode, the parameters of the confidence network get updated using instances from training set $\mathcal{D}_s$. As depicted in Figure 5.1b, each training instance is passed through the representation layer mapping inputs to vectors. These vectors are concatenated with their corresponding weak labels $\tilde{y}_j$ generated by the weak annotator. The confidence network, which is a fully connected feedforward


Prediction loss wrt. the weak labels
$\mathcal{L}_c = \sum_{j \in \mathcal{D}_s} -c_j \log(\tilde{c}_j) - (1 - c_j)\log(1 - \tilde{c}_j)$
$\{\tilde{c}_j\}_{j \in \mathcal{D}_s}$
$\{\tilde{c}_i\}_{i \in \mathcal{D}_w}$
$\{\hat{y}_i\}_{i \in \mathcal{D}_w}$
Supervision Layer
Confidence Network
$\{c_j\}_{j \in \mathcal{D}_s}$
Representation Learning
Goodness of the Sample
$\{\tilde{y}_i\}_{i \in \mathcal{D}_w}$
$\{\tilde{y}_j\}_{j \in \mathcal{D}_s}$
$\{y_j\}_{j \in \mathcal{D}_s}$
$\{x_j\}_{j \in \mathcal{D}_s}$
Weak Annotator
True Labels

(a) Full Supervision Mode: Training on batches with strong labels.


Prediction loss wrt. the weak labels
$\mathcal{L}_c = \sum_{j \in \mathcal{D}_s} -c_j \log(\tilde{c}_j) - (1 - c_j)\log(1 - \tilde{c}_j)$
$\{\tilde{c}_j\}_{j \in \mathcal{D}_s}$
$\{\tilde{c}_i\}_{i \in \mathcal{D}_w}$
$\{\hat{y}_i\}_{i \in \mathcal{D}_w}$
Supervision Layer
Confidence Network
$\{c_j\}_{j \in \mathcal{D}_s}$
Representation Learning
Goodness of the Sample
$\{\tilde{y}_i\}_{i \in \mathcal{D}_w}$
$\{\tilde{y}_j\}_{j \in \mathcal{D}_s}$
$\{y_j\}_{j \in \mathcal{D}_s}$
$\{x_i\}_{i \in \mathcal{D}_w}$
Weak Annotator
True Labels

(b) Weak Supervision Mode: Training on batches with weak labels.

Figure 5.1: Learning from controlled weak supervision: Our proposed multi-task network for learning a target task in a semi-supervised fashion, using a large amount of weakly labeled data and a small amount of data with strong labels. Faded parts of the network are disabled during the training in the corresponding mode. Red-dotted arrows show gradient propagation. Parameters of the parts of the network in red frames get updated in the backward pass, while parameters of the network in blue frames are fixed during training. (Best viewed in color.)

network with sigmoid as the output layer, estimates $\tilde{c}_j$ that is the probability of taking data instance j into account for training the target network.

In the *weak supervision* mode, the parameters of the target network are updated using the training set $\mathcal{D}_w$. As shown in Figure 5.1a, each training instance is passed through the same representation learning layer and is then processed by the supervision layer which is a part of the target network predicting the label for the main task. We also pass the learned representation of each training instance along with its corresponding label generated by the weak annotator to the confidence network to estimate the *confidence score* of the training sample, i.e., $\tilde{c}_i$. The confidence score is computed for each sample from the training set $\mathcal{D}_w$. These confidence scores are used to weight the gradient updating target network parameters or in other words the step size during back-propagation.

It is noteworthy that the representation layer is shared between both networks. Thus, besides the regularization effect of updating the parameters of this layer with respect to two loses, sharing this layer helps the confidence network to use the representation learned based on a samples of the set $\mathcal{D}_w$ as a rather large set, as well as the target network to utilize the information from samples of the set $\mathcal{D}_s$, as a rather clean set. Most importantly, sharing this layer lets the confidence network and the target network to have the same point of view on data points.

5.2.2 Training the Learner and the Meta-Learner

Here, we explain how we train CWS in which we jointly update the parameters of the target network, the learner and the confidence network, the meta-learner. Our optimization objective is composed of two terms: (1) the confidence network loss $\mathcal{L}_c$, which captures the quality of the output from the confidence network and (2) the target network loss $\mathcal{L}_t$, which expresses the quality for the main task.

Both networks are trained by alternating between the *weak supervision* and the *full supervision* mode:

Full Supervision Mode: in this mode, the parameters of the confidence network are updated using training instances drawn from training set $\mathcal{D}_s$. We use the cross-entropy loss function for the confidence network to capture the difference between the predicted confidence score of sample j, i.e., $\tilde{c}_j$ and the target score c_j:

$$\mathcal{L}_c = \sum_{j \in \mathcal{D}_s} -c_j \log(\tilde{c}_j) - (1 - c_j) \log(1 - \tilde{c}_j). \tag{5.2}$$

The target score c_j indicates how similar the strong and the weak labels are, and it is calculated with respect to the main task.

Weak Supervision Mode: In this mode, the parameters of the target network are updated using training instances from $\mathcal{D}_w$. We use a weighted loss function, $\mathcal{L}_t$, to capture the difference between the predicted label $\hat{y}_i$ by the target network and target label $\tilde{y}_i$:

$$\mathcal{L}_t = \sum_{i \in U} \tilde{c}_i \mathcal{L}_i, \tag{5.3}$$

where $\mathcal{L}_i$ is the task-specific loss on training sample i and $\tilde{c}_i$ is the confidence score of the weakly annotated sample i, estimated by the confidence network. Note that $\tilde{c}_i$ is treated as a constant during the weak supervision mode and there is no gradient propagation to the confidence network in the backward pass (as depicted in Figure 5.1a).

We minimize two loss functions jointly by randomly alternating between full and weak supervision modes (for example, using a 1:10 ratio). During training and based on the chosen supervision mode, we sample a batch of training instances from $\mathcal{D}_s$ with replacement or from $\mathcal{D}_w$ without replacement (since $\mathcal{D}_w$ can be very large). Since in our setups usually $|\mathcal{D}_w| \gg |\mathcal{D}_s|$, the training process oversamples the instance from $\mathcal{D}_s$.

The key point here is that the "main task" and "confidence scoring" task are always defined to be close tasks and sharing representation will benefit the confidence network as a kind of implicit data augmentation to compensate the small amount of data with strong labels. Besides, updating the representation layer with respect to the loss of the other network acts as a regularization for each of these networks and helps generalization for both target and confidence network and consequently less chance for overfitting.

In this section, we introduced an approach that can meta-learn the quality of labels as confidence scores, jointly with the main task at hand, when learning with weakly labeled samples that have variable qualities. We showed that we can incorporate the estimated confidence scores associated with each weakly labeled sample to control the magnitude of the parameter updates during training based on the quality of that sample. In the next section, we introduce another approach that can estimate the quality of samples and regulate the learning process based on it.

5.3 Fidelity-Weighted Learning

In this section, we address the second research question of this chapter:

> **RQ-2.2.2** *When learning from samples of variable quality, can we reannotate these samples and provide (hopefully) better labels, associated with a fidelity score to regulate the learning rate?*

We introduce Fidelity-Weighted Learning (FWL), a Bayesian semi-supervised approach that leverages a small amount of data with strong labels to generate a larger training set with *fidelity-weighted weakly-labeled samples*, which can then be used to modulate the learning process based on the quality of each weak sample. By directly modeling the inaccuracies introduced by the weak annotator in this way, we can control the extent to which we make use of this additional source of weak supervision: more for confidently-labeled weak samples close to the true observed data, and less for uncertain samples further away from the observed data. We use a non-parametric kernel-based method to measure the closeness.

We propose a setting consisting of two main modules. One is called the student and is in charge of learning a suitable data representation and performing the main prediction task (similar to the target network in Section 5.2), the other is the teacher which modulates the learning process by modeling the inaccuracies in the labels.

Similar to CWS (introduced in Section 5.2), FWL learns to modulate the learning process based on quality of labels. However, unlike CWS, FWL operates in three different sequential stages and not only it learns to weight the samples based on their quality, but also it learns re-estimate a better labels, i.e. correct the weak labels, during training.

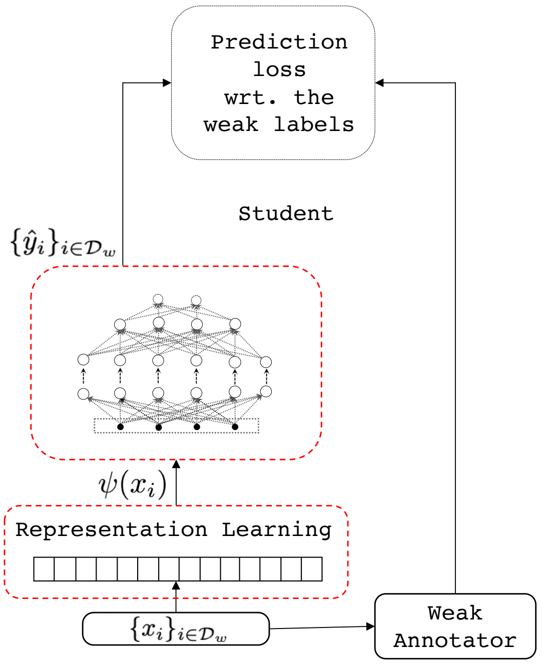
5.3.1 Recipe of the Fidelity-Weighted Learning

In this section, we describe the FWL approach for semi-supervised learning when we have access to weak supervision (e.g., heuristics or weak annotators).

Our proposed setup comprises a neural network called the **student** and a Bayesian function approximator called the **teacher**. The training process consists of three phases which we summarize in Algorithm 5.1 and Figure 5.2.

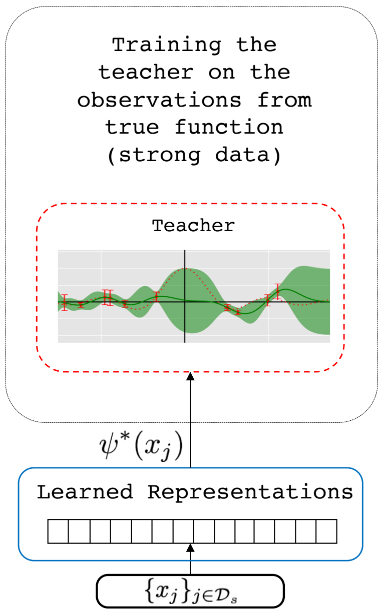
Step 1 *Pre-train the student on $\mathcal{D}_w$ using weak labels generated by the weak annotator.* (See Figure 5.2b.)

The main goal of this step is to learn a *task dependent* representation of the data as well as pretraining the student. The student function is a neural network consisting of two parts. The first part $\psi(\cdot)$ learns the data representation and the second part $\phi(\cdot)$ performs the prediction task (e.g., classification). Therefore the overall function is $\hat{y} = \phi(\psi(x_i))$. The student is trained on all samples of the weak dataset $\mathcal{D}_w = \{(x_i, \tilde{y}_i)\}_{i=1}^{M}$. For brevity, in the following, we will refer to both data sample x_i and its representation $\psi(x_i)$ by x_i when it is obvious from the context. From self-supervised feature learning point of view, we can say that representation learning in this step is solving a surrogate task of approximating the expert knowledge, for which a noisy supervision signal is provided by the weak annotator.



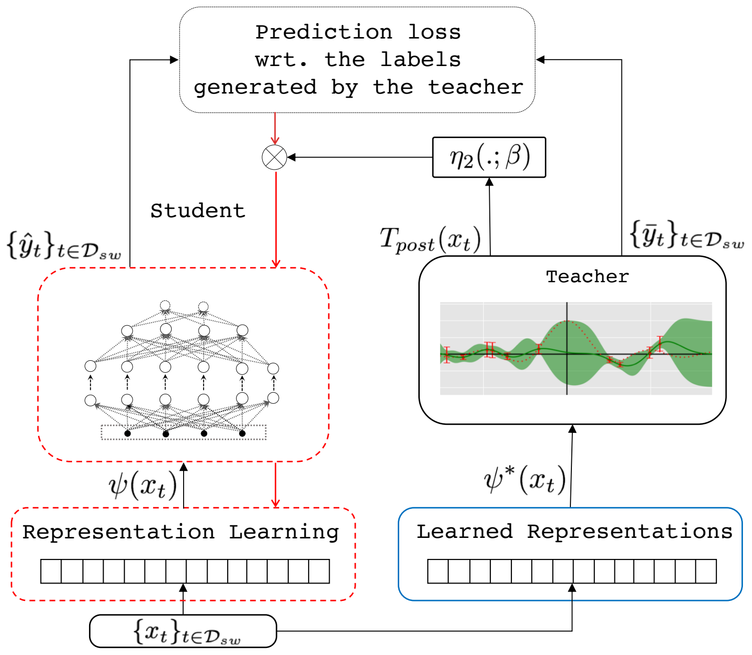
Prediction loss wrt. the weak labels
Student
$\{\hat{y}_i\}_{i \in \mathcal{D}_w}$
$\psi(x_i)$
Representation Learning
$\{x_i\}_{i \in \mathcal{D}_w}$
Weak Annotator

(a) Step 1



Training the teacher on the observations from true function (strong data)
Teacher
$\psi^*(x_j)$
Learned Representations
$\{x_j\}_{j \in \mathcal{D}_s}$

(b) Step 2



Prediction loss wrt. the labels generated by the teacher
$\eta_2(.;\beta)$
Student
$\{\hat{y}_t\}_{t \in \mathcal{D}_{sw}}$
$T_{post}(x_t)$
$\{\bar{y}_t\}_{t \in \mathcal{D}_{sw}}$
Teacher
$\psi(x_t)$
$\psi^*(x_t)$
Representation Learning
Learned Representations
$\{x_t\}_{t \in \mathcal{D}_{sw}}$

(c) Step 3

Figure 5.2: Illustration of Fidelity-Weighted Learning: Step 1: Pre-train student on weak data, Step 2: Fit teacher to observations from the true function, and Step 3: Fine-tune student on labels generated by teacher, taking the confidence into account. Red dotted borders and blue solid borders depict components with trainable and non-trainable parameters, respectively.

Step 2 *Train the teacher on the strong data $(\psi(x_j), y_j) \in \mathcal{D}_s$ represented in terms of the student representation $\psi(\cdot)$ and then use the teacher to generate a soft dataset $\mathcal{D}_{sw}$ consisting of $\langle$sample, predicted label, confidence$\rangle$ for **all** data samples.* (See Figure 5.2b.)

We use a Gaussian process ($\mathcal{GP}$) as the teacher to capture the label uncertainty in terms of the student representation, estimated w.r.t. the strong data. A prior mean and co-variance function is chosen for $\mathcal{GP}$. The learned embedding function $\psi(\cdot)$ in Step 1 is then used to map the data samples to dense vectors as input to the $\mathcal{GP}$. We use the learned representation by the student in the previous step to compensate lack of data in $\mathcal{D}_s$ and the teacher can enjoy the learned knowledge from the large quantity of the weakly annotated data. This way, we also let the teacher see the data through the lens of the student.

The $\mathcal{GP}$ is trained on the samples from $\mathcal{D}_s$ to learn the posterior mean $\boldsymbol{m}_{\text{post}}$ (used to generate soft labels) and posterior co-variance $K_{\text{post}}(\cdot,\cdot)$ (which represents label uncertainty). We then create the *soft dataset* $\mathcal{D}_{sw} = \{(x_t, \bar{y}_t)\}_{t=1}^{|\mathcal{D}_w \cup \mathcal{D}_s|}$, using the posterior $\mathcal{GP}$, input samples x_t from $\mathcal{D}_w \cup \mathcal{D}_s$, and predicted labels $\bar{y}_t$ with their associated uncertainties as computed by $T(x_t)$ and $\Sigma(x_t)$:

$$\begin{aligned} T(x_t) &= g(\boldsymbol{m}_{\text{post}}(x_t)) \\ \Sigma(x_t) &= h(K_{\text{post}}(x_t, x_t)) \end{aligned}$$

The re-generated labels, $\bar{y}_t$, which are called *soft labels*, are equal to strong labels y_t, when $x_t \in \mathcal{D}_s$ (with zero uncertainty), and when $x_t \in \mathcal{D}_w$, $\bar{y}_t$ are supposed to be a better labels than the original weak labels $\tilde{y}_t$ (with an uncertainty that is estimated by the $\mathcal{GP}$). $g(\cdot)$ transforms the output of $\mathcal{GP}$ to the suitable output space. For example, in classification tasks, $g(\cdot)$ would be the softmax function to produce probabilities that sum up to one. For multidimensional-output tasks where a vector of variances is provided by the $\mathcal{GP}$, the vector $K_{\text{post}}(x_t, x_t)$ is passed through an aggregating function $h(\cdot)$ to generate a scalar value for the uncertainty of each sample. Note that we train $\mathcal{GP}$ only on the strong dataset $\mathcal{D}_s$ but then use it to generate soft labels $\bar{y}_t = T(x_t)$ and uncertainty $\Sigma(x_t)$ for samples belonging to $\mathcal{D}_{sw} = \mathcal{D}_w \cup \mathcal{D}_s$.

In practice, we furthermore divide the space of data into several regions and assign each region a separate $\mathcal{GP}$ trained on samples from that region. This leads to a better exploration of the data space and makes use of the inherent structure of data. The resulting algorithm, called clustered $\mathcal{GP}$, gave better results compared to a single $\mathcal{GP}$. We describe the detail of the clustered $\mathcal{GP}$ in Section 5.3.2.

By this division of space, we take advantage of the knowledge learned by several teachers, each an expert on its specific region of the data space, which helps in particular when the dimensionality of the input is rather high. As a nice side-effect, this also solves the scalability issues of $\mathcal{GP}$s in that we can increase

Algorithm 5.1 Fidelity-Weighted Learning.

```
1: Train the student on samples from the weakly-annotated data $D_w$.
2: Freeze the representation-learning component $\psi(.)$ of the student and train teacher
   on the strong data $\mathcal{D}_s = \{(\psi(x_j), y_j)\}_{j=1}^N$. Apply teacher to unlabeled samples $x_t$
   to obtain soft dataset $\mathcal{D}_{sw} = \{(x_t, \bar{y}_t)\}_{t=1}^{|\mathcal{D}_w \cup \mathcal{D}_s|}$ where $\bar{y}_t = T(x_t)$ is the soft label and
   for each instance $x_t$, the uncertainty of its label, $\Sigma(x_t)$, is provided by the teacher.
3: Train the student on samples from $\mathcal{D}_{sw}$ with SGD and modulate the step-size $\eta_t$
   according to the per-sample quality estimated using the teacher (Equation 5.4).
```

the number of regions until the number of points in each region is tractable with a single $\mathcal{GP}$, and train these models in parallel.

Step 3 *Fine-tune the weights of the student network on the soft dataset, while modulating the magnitude of each parameter update by the corresponding teacher-confidence in its label.* (See Figure 5.2b.)

The student network of Step 1 is fine-tuned using samples from the soft dataset $\mathcal{D}_{sw} = \{(x_t, \bar{y}_t)\}_{t=1}^{|\mathcal{D}_w \cup \mathcal{D}_s|}$ where $\bar{y}_t = T(x_t)$. The corresponding uncertainty $\Sigma(x_t)$ of each sample is mapped to a confidence value according to Equation 5.4 below, and this is then used to determine the step size for each iteration of the stochastic gradient descent (SGD). So, intuitively, for data points where we have true labels, the uncertainty of the teacher is almost zero, which means we have high confidence and a large step-size for updating the parameters. However, for data points where the teacher is not confident, we down-weight the training steps of the student. This means that at these points, we keep the student function as it was trained on the weak data in Step 1.

More specifically, we update the parameters of the student by training on $\mathcal{D}_{sw}$ using SGD:

$$\begin{aligned}
\boldsymbol{w}^* &= \operatorname*{arg\,min}_{\boldsymbol{w} \in \mathcal{W}} \frac{1}{N} \sum_{(x_t, \bar{y}_t) \in \mathcal{D}_{sw}} l(\boldsymbol{w}, x_t, \bar{y}_t) + \mathcal{R}(\boldsymbol{w}), \\
\boldsymbol{w}_{t+1} &= \boldsymbol{w}_t - \eta_t(\nabla l(\boldsymbol{w}, x_t, \bar{y}_t) + \nabla \mathcal{R}(\boldsymbol{w}))
\end{aligned}$$

where $l(\cdot)$ is the per-sample loss, η_t is the total learning rate, N is the size of the soft dataset $\mathcal{D}_{sw}$, $\boldsymbol{w}$ is the parameters of the student network, and $\mathcal{R}(\cdot)$ is the regularization term.

We define the total learning rate as $\eta_t = \eta_1(t)\eta_2(x_t)$, where $\eta_1(t)$ is the usual learning rate of our chosen optimization algorithm that anneals over training iterations, and $\eta_2(x_t)$ is a function of the label uncertainty $\Sigma(x_t)$ that is computed by the teacher for each data point. Multiplying these two terms gives us the total learning rate. In other words, η_2 represents the *fidelity* (quality) of the current sample, and is used to multiplicatively modulate η_1. Note that η_1 does not necessarily depend on each data point, whereas η_2 does. We propose

$$\eta_2(x_t) = \exp[-\beta\Sigma(x_t)], \tag{5.4}$$

to exponentially decrease the learning rate for data point x_t if its corresponding soft label $\bar{y}_t$ is unreliable (far from a true sample). In practice, when using mini-batches, we implement this by multiplying the loss of each sample in the batch by its fidelity score and averaging over these fidelity-weighted losses in the batch when calculating the batch gradient based on that loss. In Equation 5.4, β is a positive scalar hyper-parameter. Intuitively, a small β results in a student that listens more carefully to the teacher and copies its knowledge, while a large β makes the student pay less attention to the teacher, staying with its initial weak knowledge.

5.3.2 Multi-Teacher FWL using Clustered GP

In this section, we explain the clustered GP which is an effective way of applying $\mathcal{GP}$where the scale of the data increases. Clustered GP suggests using several $\mathcal{GP} = \{GP_{c_i}\}$ to explore the entire data space more effectively. Even though inducing points and stochastic methods make $\mathcal{GP}$s more scalable we still observed poor performance when the entire dataset was modeled by a single $\mathcal{GP}$. Therefore, the reason for using multiple $\mathcal{GP}$s is mainly empirical inspired by [255] which is explained in the following:

We used Sparse Gaussian Process implemented in GPflow. The algorithm is scalable in the sense that it is not $O(N^3)$ as the original $\mathcal{GP}$ is, but it introduces inducing points in the data space and defines a variational lower bound for the marginal likelihood. The variational bound can be optimized by stochastic methods, which makes the algorithm applicable in large datasets. However, the tightness of the bound depends on the location of the inducing points, which are found through the optimization process.

In [79] (Appendix A), it is empirically observed that a single $\mathcal{GP}$ does not give a satisfactory accuracy on left-out test dataset on our tasks/datasets. This can be due to the inability of the algorithm to find good inducing points when the number of inducing points is restricted to just a few. Then we increased the number of inducing points M which trades off the scalability of the algorithm because it scales with $O(NM^2)$. Moreover, apart from scalability which is partly solved by stochastic methods, we argue that the structure of the entire space may not be explored well by a single $\mathcal{GP}$ and its inducing points. This can be due to the observation that our datasets are distributed in a highly sparse way within the high dimensional embedding space. To cure the problem, one can use PCA to reduce input dimensions and give a denser representation, but based on the experiments in [79], it does not result in a considerable improvement.

We may be able to argue that clustered $\mathcal{GP}$ makes better use of the data structure roughly close to the idea of KISS-GP [305]. In inducing point methods, it is normally assumed that $k \ll h$ (k is the number of inducing points and h is the number of training samples) for computational and storage saving. However, we have this intuition that few number of inducing points make the model unable to explore the inherent structure of data [305]. By employing several GPs, we were able to use a large number of inducing points even when $k > h$ which seemingly better exploits the structure of datasets. Because our work was not aimed to be a close investigation of GP, we considered clustered $\mathcal{GP}$ as the engineering side of the work which is a tool to give us a measure of confidence. Other tools such as a single $\mathcal{GP}$ with inducing points that form a Kronecker or Toeplitz covariance matrix are also conceivable. Therefore, we do not of course claim that we have proposed a new method of inference for GPs. Here is practical description of clustered $\mathcal{GP}$ algorithm: *Clustered* $\mathcal{GP}$: Let N be the size of the dataset on which we train the teacher. Assume we allocate K teachers to the entire data space. Therefore, each $\mathcal{GP}$ sees a dataset of size $n = N/K$. Then we use a simple clustering method (e.g., k-means) to find centroids of K clusters $C_1, C_2, \ldots, C_K$ where C_i consists of samples $\{x_{i,1}, x_{i,2}, \ldots, x_{i,n}\}$. We take the centroid c_i of cluster C_i as the representative sample for all its content. Note that c_i does not necessarily belong to $\{x_{i,1}, x_{i,2}, \ldots, x_{i,n}\}$. We assign each cluster a $\mathcal{GP}$ trained by samples belonging to that cluster. More precisely, cluster C_i is assigned a $\mathcal{GP}$ whose data points are $\{x_{i,1}, x_{i,2}, \ldots, x_{i,n}\}$. Because there is no dependency among different clusters, we train them in parallel to speed-up the procedure.

The pseudo-code of the clustered $\mathcal{GP}$ is presented in Algorithm 5.2. When the main issue is computational resources (when the number of inducing points for each $\mathcal{GP}$ is large), we can first choose the number n which is the maximum size of the dataset on which our resources allow us to train a $\mathcal{GP}$, then find the number of clusters $K = N/n$ accordingly. The rest of the algorithm remains unchanged.

5.3.3 FWL on a Toy Example

To better understand FWL, we apply FWL to a one-dimensional toy problem to illustrate the various steps. Let $f_t(x) = \sin(x)$ be the true function (red dotted line in Figure 5.3a) from which a small set of observations $\mathcal{D}_s = \{(x_j, y_j)\}_{j=1}^{N}$ is provided (red points in Figure 5.3b). These observation might be noisy, in the same way that labels obtained from a human labeler could be noisy. A weak annotator function $f_w(x) = 2sinc(x)$ (magenta line in Figure 5.3a) is provided, as an approximation to $f_t(\cdot)$.

The task is to obtain a good estimate of $f_t(\cdot)$ given the set $\mathcal{D}_s$ of strong obser-

Algorithm 5.2 Clustered Gaussian processes.

```
1: Let $N$ be the sample size, $n$ the sample size of each cluster, $K$ the number of clusters, and $c_i$ the center of cluster $i$.
2: Run K-means with $K$ clusters over all samples with true labels $\mathcal{D}_s = \{(x_j, y_j)\}_{j=1}^{N}$.
       $\text{K-means}(x_j) \rightarrow c_1, c_2, \ldots, c_K$
   where $c_i$ represents the center of cluster $C_i$ containing samples $D_s^{c_i} = \{x_{i,1}, x_{i,2}, ...x_{i,n}\}$.
3: Assign each of $K$ clusters a Gaussian process and train them in parallel to approximate the label of each sample:
       $\mathcal{GP}_{c_i}(\boldsymbol{m}_{\text{post}}^{c_i}, K_{\text{post}}^{c_i}) = \mathcal{GP}(\boldsymbol{m}_{\text{prior}}, K_{\text{prior}})|D_s^{c_i} = \{(\psi(x_{s,c_i}), y_{s,c_i})\}$
       $T_{c_i}(x_t) = g(\boldsymbol{m}_{\text{post}}^{c_i}(x_t))$
       $\Sigma_{c_i}(x_t) = h(K_{\text{post}}^{c_i}(x_t, x_t))$
   ,
4: where $\mathcal{GP}_{c_i}$ is trained on $\mathcal{D}_s^{c_i}$ containing samples belonging to the cluster $c_i$. Other elements are defined in Section 5.3.1
5: Use trained teacher $T_{c_i}(.)$ to evaluate the soft label and uncertainty for samples from $\mathcal{D}_{sw}$ to compute $\eta_2(x_t)$ required for step 3 of Algorithm 5.1. We use $T(.)$ as a wrapper for all teachers $\{T_{c_i}\}$.
```

vations and the weak annotator function $f_w(\cdot)$. We can easily obtain a large set of observations $\mathcal{D}_w = \{(x_i, \tilde{y}_i)\}_{i=1}^{M}$ from $f_w(\cdot)$ with almost no cost (magenta points in Figure 5.3a).

As the teacher, we use standard Gaussian process regression[2] with this kernel:

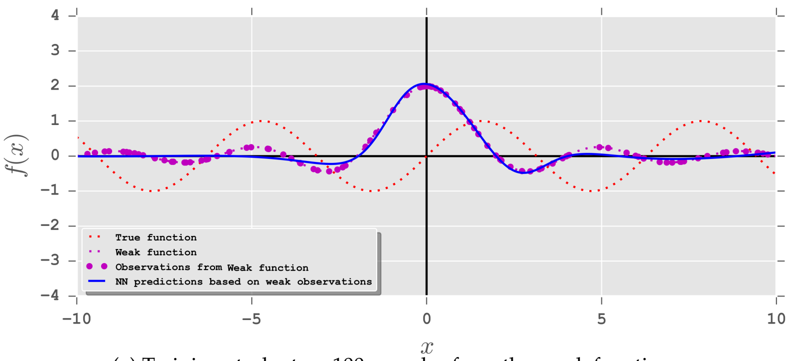
$$k(x_i, x_j) = k_{\text{RBF}}(x_i, x_j) + k_{\text{White}}(x_i, x_j), \tag{5.5}$$

where

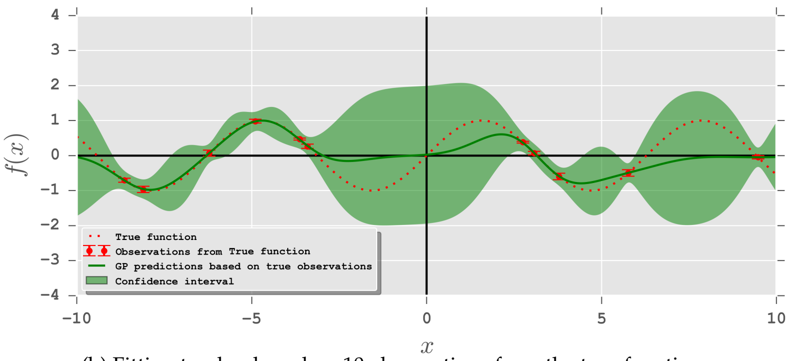
$$k_{\text{RBF}}(x_i, x_j) = \exp\left(\frac{\|x_i - x_j\|^2}{2^2}\right),$$
$$k_{\text{White}}(x_i, x_j) = constant\_value \ \forall x_i = x_j, \text{ and 0 otherwise.}$$

We fit only one $\mathcal{GP}$ on all the data points (i.e., no clustering). Also during fine tuning, we set $\beta = 1$. The student is a simple feed-forward network with the depth of 3 layers and width of 128 neurons per layer. We have used *tanh* as the nonlinearity for the intermediate layers and a linear output layer. As the optimizer, we used Adam [161] and the initial learning rate has been set to 0.001.

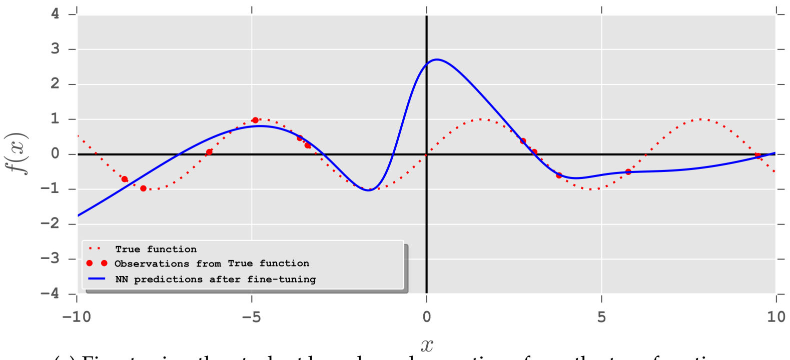
[2] *http://gpflow.readthedocs.io/en/latest/notebooks/regression.html*



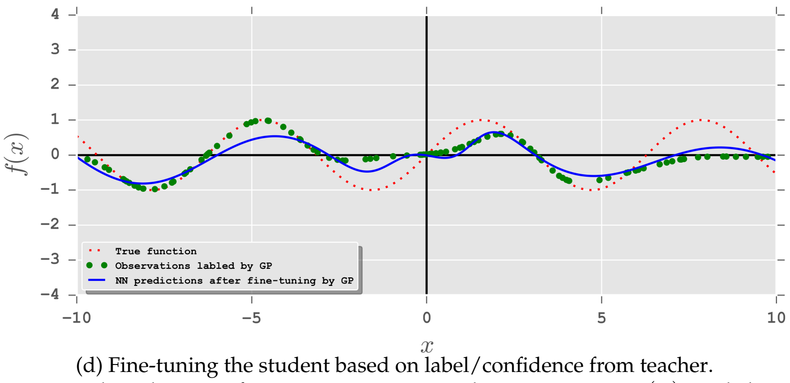
(a) Training student on 100 samples from the weak function.



(b) Fitting teacher based on 10 observations from the true function.



(c) Fine-tuning the student based on observations from the true function.



(d) Fine-tuning the student based on label/confidence from teacher.

Figure 5.3: Toy example: The true function we want to learn is $y = \sin(x)$ and the weak function is $y = 2sinc(x)$.

We randomly sample 100 data points from the weak annotator and 10 data points from the true function. We introduce a small amount of noise to the observation of the true function to model the noise in the human labeled data.

We consider two experiments:

1. A neural network trained on weak data and then fine-tuned on strong data from the true function, which is the most common semi-supervised approach (Figure 5.3c).
2. A teacher-student framework working by the proposed FWL approach.

As can be seen in Figure 5.3d, by taking into account label confidence, FWL gives a better approximation of the true hidden function, compared to the standard fine tuning. We repeated the above experiment 10 times. The average RMSE with respect to the true function on a set of test points over those 10 experiments for the student, were as follows:

1. Student is trained on weak data (blue line in Figure 5.3a): 0.8406,
2. Student is trained on weak data then fine tuned on true observations (blue line in Figure 5.3c): 0.5451,
3. Student is trained on weak data, then fine tuned by soft labels and confidence information provided by the teacher (blue line in Figure 5.3d): 0.4143 (best).

5.4 Applying CWS and FWL on Language Understanding Tasks

We apply CWS and FWL, two approaches introduced in this chapter for learning from a vast amount of weakly annotated data, while a small set of labeled data exist, to two different tasks: *document ranking* and *sentiment classification*. Whilst these two applications differ considerably, as do the exact operationalizations of the proposed models to these cases, in both cases the human gold standard data is based on a cognitively complex, or subjective, judgments causing high interrater variation, increasing both the cost of obtaining labels and the need for larger sets of labels.

For both tasks, we evaluate the performance of CWS as well as FWL compared to some baselines that are described in Table 5.1. In the rest of this chapter, we present results of different experiments and studies and we refer to these baselines using their id and name (first and the second column in Table 5.1).

[3] In the document ranking task, as the representation of documents and queries we use weighted averaging over pretrained embeddings of their words based on their inverse document frequency [84]. In the sentiment analysis task, we use skip-thoughts vectors [164].

Table 5.1: Descriptions of baseline models.

		Basic Baselines
1	**WA**	The weak annotator, i.e., the unsupervised method used for annotating the unlabeled data.
2	$\mathbf{NN_S}$	Full Supervision Only, i.e., the target network (or the student) trained only on strong labeled data ($\mathcal{D}_s$).
3	$\mathbf{NN_W}$	Weak Supervision Only, i.e., the target network (or the student) trained only on weakly labeled data ($\mathcal{D}_w$).
4	$\mathbf{NN_{W/S^+}}$	Weak Supervision + Oversampled Strong Supervision, i.e., the target network (or the student) trained on samples that are alternately drawn from $\mathcal{D}_w$ without replacement, and $\mathcal{D}_s$ with replacement. Since $\|\mathcal{D}_s\| \ll \|\mathcal{D}_w\|$, it oversamples the strong data.
5	$\mathbf{NN_W \rightarrow NN_S}$	Weak Supervision + Fine Tuning, i.e., the target network (or the student) trained on weak dataset $\mathcal{D}_w$ and fine-tuned on strong dataset $\mathcal{D}_s$.
6	$\mathbf{NN_W \rightarrow NN_S^{Sup}}$	Weak Supervision + Supervision Layer fine tuning, i.e., the target network (or the student) trained only on on weak dataset $\mathcal{D}_w$ and the supervision layer is fine-tuned on strong dataset $\mathcal{D}_s$, while the representation learning layer is fixed.
7	$\mathbf{NN_W \rightarrow NN_S^{Rep}}$	Weak Supervision + Representation Learning Layer Fine Tuning, i.e., the target network (or the student) trained only on on weak dataset $\mathcal{D}_w$ and the representation layer is fine-tuned on strong dataset $\mathcal{D}_s$, while the representation learning layer is fixed.
		Controlled Weak Supervision
8	**CWS**	Learning from Controlled Weak Supervision as explained in Section 5.2.
9	$\mathbf{CWS_{JT+}}$	Controlled Weak Supervision with Joint Training is the same as CWS (explained in Section 5.2.2), except that parameters of the supervision layer in target network are also updated using batches from V, with regards to the strong labels.
10	$\mathbf{CWS_{ST}}$	Separate Training, i.e., we consider the confidence network as a separate network, without sharing the representation learning layer, and train it on set V. We then train the target network on the controlled weak supervision signals.
11	$\mathbf{CWS_{CT}}$	Circular Training, i.e., we train the target network on set U. Then the confidence network is trained on data with strong labels, and the target network is trained again but on controlled weak supervision signals.
12	$\mathbf{CWS_{PT}}$	Progressive Training is the mixture of the two previous baselines. Inspired by [242], we transfer the learned information from the converged target network to the confidence network using progressive training. We then train the target network again on the controlled weak supervision signals.
		Fidelity Weighted Learning
13	**FWL**	Fidelity Weighted Learning that is explained in Section 5.3.
14	$\mathbf{NN_{W^{\omega} \rightarrow NN_S}}$	The student trained on the weak data, but the step-size of each weak sample is weighted by a fixed value $0 \leq \omega \leq 1$, and fine-tuned on strong data. As an approximation for the optimal value for ω, we have used the mean of η_2 of our model (below).
15	$\mathbf{FWL}_{unsuprep}$	The representation in the first step is trained in an unsupervised way[3] and the student is trained on samples labeled by the teacher using the confidence scores.
16	$\mathbf{FWL \backslash \Sigma}$	The student trained on the weakly labeled data and fine-tuned on samples labeled by the teacher without taking the confidence into account.

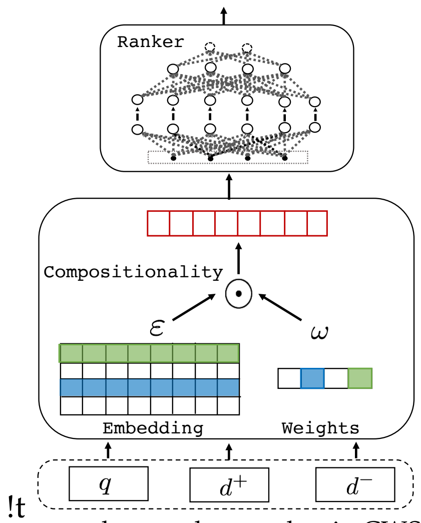


Figure 5.4: The document ranker used as teacher in CWS and student in FWL.

5.4.1 Document Ranking

This task is a core information retrieval problem and is challenging as the ranking model needs to learn a representation for long documents and capture the notion of relevance between queries and documents. Furthermore, as it was discussed in Chapter 4, the size of publicly available datasets with query-document relevance judgments is unfortunately quite small ($\sim$250 queries). In our experiments, ranking is cast as a regression task. Given each training sample x as a triple of query q, and two documents d^+ and d^-, the goal is to learn a function $\mathcal{F} : \{\langle q, d^+, d^- \rangle\} \rightarrow \mathbb{R}$, which maps each data sample x to a scalar output value y indicating the probability of d^+ being ranked higher than d^- with respect to q.

The target network in CWS and the student in FWL

We employ the pairwise neural ranker architecture explained in Section 4.2.2 as the target network in CWS and student in FWL.

Each training instance x consists of a query q, and two documents d^+ and d^-. The labels, $\tilde{y}$ and y, are scalar values indicating the probability of d^+ being ranked higher than d^- with respect to q.

The Representation Learning Layer. This layer learns a function $\varepsilon : \mathcal{V} \rightarrow \mathbb{R}^m$ (where $\mathcal{V}$ denotes the vocabulary set, and m is the dimension of embedding) that maps each word to its embedding as well as a weighting function $\omega : \mathcal{V} \rightarrow \mathbb{R}$ which learns the global importance of each word. Then, the learned weights are used to compose word embeddings to generate query/document embeddings. The output of this layer is the concatenation of vectors representing query and two documents. In our experiments, we initialize the embedding function ε with word2vec embeddings [195] pre-trained on Google News and the weighting function ω with IDF.

The Supervision Layer. This layer receives the vector representation of the inputs processed by the representation learning layer and outputs a prediction $\hat{y}_i$. We opt for a simple fully connected feed-forward network with l hidden layers followed by a sigmoid. We employ the weighted cross entropy loss:

$$\mathcal{L}_t = \sum_{i \in B_U} \tilde{c}_i[-\tilde{y}_i \log(\hat{y}_i) - (1 - \tilde{y}_i)\log(1 - \hat{y}_i)], \tag{5.6}$$

where $B_{\mathcal{D}_w}$ is a batch of samples from $\mathcal{D}_w$, and $\tilde{c}_i$ is the confidence score of the weakly annotated sample i, estimated by the confidence network.

The general schema of the target network (or student) is illustrated in Figure 5.4. More details are provided in Section 4.2.2.

The weak annotator

The weak annotator in the document ranking task is BM25 [232], a well-known unsupervised retrieval method. This method heuristically scores a given pair of query-document based on the statistics of their matched terms. In the pairwise document ranking setup, $\tilde{y}_i$ for a given sample $x_j = (q, d^+, d^-)$ is the probability of document d^+ being ranked higher than d^-: $\tilde{y}_i = P_{q,d^+,d^-} = {}^{s_{q,d^+}}/{}_{s_{q,d^+}+s_{q,d^-}}$, where $s_{q,d}$ is the score obtained from the weak annotator.

The confidence network in CWS

The confidence network is a regressor and we use a simple fully connected feed-forward network. To train the confidence network, the target label c_j is calculated using the absolute difference of the strong label and the weak label: $c_j = 1 - |y_j - \tilde{y}_j|$, where y_j is calculated similar to $\tilde{y}_i$, but $s_{q,d}$ comes from strong labels provided by humans.

The teacher in FWL

We use Gaussian Process as the teacher in order to generate soft labels. We pass the mean of $\mathcal{GP}$ through the same function $g(\cdot)$ that is applied on the output of the student network, where the $g(\cdot)$ is sigmoid for the document ranking task. Since we have one dimensional regression here, $\Sigma(x_t)$ is scalar and $h(\cdot)$ is identity. In the teacher, linear combinations of different kernels are used. For the document ranking task, we use sparse variational GP regression[4] [279] with this kernel:

$$k(x_i, x_j) = k_{\text{Matern3/2}}(x_i, x_j) + k_{\text{Linear}}(x_i, x_j) + k_{\text{White}}(x_i, x_j), \tag{5.7}$$

where

$$\begin{aligned} k_{\text{Matern3/2}}(x_i, x_j) &= \left(1 + \frac{\sqrt{3}\|x_i - x_j\|}{l}\right) \exp\left(-\frac{\sqrt{3}\|x_i - x_j\|}{l}\right), \\ k_{\text{Linear}}(x_i, x_j) &= \sigma_0^2 + x_i \cdot x_j, \\ k_{\text{White}}(x_i, x_j) &= \mathit{constant\_value}\ \forall x_i = x_j, \text{ and 0 otherwise.} \end{aligned}$$

We empirically found that $l = 1$ satisfying value for the length scale of the Matern3/2 kernel. We also set $\sigma_0 = 0$ to obtain a homogeneous linear kernel. The constant value of $K_{White}(\cdot, \cdot)$ determines the level of noise in the labels. This is different from the noise in weak labels. This term explains the fact that even in strong labels there might be a trace of noise due to the inaccuracy of human labelers. We set the number of clusters to 50 for this task in the clustered $\mathcal{GP}$ algorithm.

Collections

We use two standard TREC collections for the task of ad-hoc retrieval: The first collection (*Robust04*) consists of 500k news articles from different news agencies as a homogeneous collection. The second collection (*ClueWeb*) is ClueWeb09 Category B, a large-scale web collection with over 50 million English documents, which is considered to be a heterogeneous collection. Spam documents are filtered out using the Waterloo spam scorer[5] [57] with the default threshold 70%.

Data with strong labels. We take query sets that contain human-labeled judgments: a set of 250 queries (TREC topics 301–450 and 601–700) for the Robust04 collection and a set of 200 queries (topics 1-200) for the experiments on the ClueWeb collection. For each query, we take all documents judged as relevant plus the same number of documents judged as non-relevant and form pairwise combinations among them.

Data with weak labels. We create a query set Q using the unique queries appearing in the AOL query logs [215]. This query set contains web queries initiated by real users in the AOL search engine that were sampled from a three-month period from March 2006 to May 2006. We apply standard pre-processing [81, 84]

[4] *http://gpflow.readthedocs.io/en/latest/notebooks/SGPR_notes.html*
[5] *http://plg.uwaterloo.ca/~gvcormac/clueweb09spam/*

on the queries: We filter out a large volume of navigational queries containing URL substrings ("http", "www.", ".com", ".net", ".org", ".edu"). We also remov all non-alphanumeric characters from the queries. For each dataset, we take queries that have at least ten hits in the target corpus using our weak annotator method. Applying all these steps, we collect 6.15 million queries to train on in Robust04 and 6.87 million queries for ClueWeb. To prepare the weakly labeled training set $\mathcal{D}_w$, we take the top $1,000$ retrieved documents using BM25 for each query from training query set Q, which in total leads to $\sim|Q| \times 10^6$ training samples.

Experimental Setup

We conduct 3-fold cross-validation. However, for each dataset, we first tun all the hyper-parameters of the target network in CWS (and student in the first step of FWL) in the first step on the set with strong labels using batched GP bandits with an expected improvement acquisition function [87] and keep the optimal parameters of the target network (and student) fixed for all the other experiments. The size and number of hidden layers for the target network (and student) is selected from $\{64, 128, 256, 512\}$. The initial learning rate and the dropout parameter are selected from $\{10^{-3}, 10^{-5}\}$ and $\{0.0, 0.2, 0.5\}$, respectively. We consider embedding sizes of $\{300, 500\}$. The batch size in our experiments is set to 128. We use ReLU [203] as a non-linear activation function α in target network (and student). We use the Adam optimizer [161] for training, and dropout as a regularization technique.

At inference time, for each query, we take the top $2,000$ retrieved documents using BM25 as candidate documents and re-rank them using the trained models. We use the Indri[6] implementation of BM25 with default parameters (i.e., $k_1 = 1.2$, $b = 0.75$, and $k_3 = 1,000$).

Results and Discussion

We conduct k-fold cross validation on $\mathcal{D}_s$ (the strong data) and report two standard evaluation metrics for ranking: mean average precision (MAP) of the top-ranked $1,000$ documents and normalized discounted cumulative gain calculated for the top 20 retrieved documents (nDCG@20). Table 5.2 shows the performance on both datasets. As can be seen, FWL and CWS both provide significant boosts on the performance on top of the baseline methods over both datasets.

In the ranking task, the student is designed in particular to be trained on weak

[6]*https://www.lemurproject.org/indri.php*

Table 5.2: Performance of CWS and FWL as well as the main baseline methods, described in Table 5.1, for the document ranking task. $^{\blacktriangle i}$ indicates that the improvements with respect to the baseline i are statistically significant at the 0.05 level using the paired two-tailed t-test with Bonferroni correction.

	Method	**Robust04**		**ClueWeb**	
		MAP	*nDCG@20*	*MAP*	*nDCG@20*
1	**WA$_{\text{BM25}}$**	$0.2503^{\blacktriangle 2}$	$0.4102^{\blacktriangle 2}$	$0.1021^{\blacktriangle 2}$	$0.2070^{\blacktriangle 2}$
2	**NN$_{\text{S}}$**	0.1790	0.3519	0.0782	0.1730
3	**NN$_{\text{W}}$**	$0.2702^{\blacktriangle 12}$	$0.4290^{\blacktriangle 12}$	$0.1297^{\blacktriangle 12}$	$0.2201^{\blacktriangle 12}$
4	**NN$_{\text{W/S}^+}$**	$0.2763^{\blacktriangle 123}$	$0.4330^{\blacktriangle 123}$	$0.1354^{\blacktriangle 123}$	$0.2319^{\blacktriangle 123}$
5	**NN$_{\text{W}} \rightarrow$ NN$_{\text{S}}$**	$0.2810^{\blacktriangle 12346}$	$0.4372^{\blacktriangle 12346}$	$0.1346^{\blacktriangle 12346}$	$0.2317^{\blacktriangle 12346}$
6	**NN$_{\text{W}} \rightarrow$ NN$_{\text{S}}^{\text{Sup}}$**	$0.2711^{\blacktriangle 123}$	$0.4203^{\blacktriangle 123}$	$0.1002^{\blacktriangle 123}$	$0.1940^{\blacktriangle 123}$
7	**NN$_{\text{W}} \rightarrow$ NN$_{\text{S}}^{\text{Rep}}$**	$0.2810^{\blacktriangle 1234}$	$0.4316^{\blacktriangle 1234}$	$0.1286^{\blacktriangle 1234}$	$0.2240^{\blacktriangle 1234}$
8	**CWS**	$0.3017^{\blacktriangle 1234567}$	$0.4511^{\blacktriangle 1234567}$	$0.1363^{\blacktriangle 1234567}$	$0.2444^{\blacktriangle 1234567}$
13	**FWL**	$\mathbf{0.3124}^{\blacktriangle 1234567}$	$\mathbf{0.4600}^{\blacktriangle 1234567}$	$\mathbf{0.1472}^{\blacktriangle 1234567}$	$\mathbf{0.2453}^{\blacktriangle 1234567}$

annotations [84], hence training the network only on weak supervision, i.e., NN$_{\text{W}}$ performs better than NN$_{\text{S}}$. This may be due to the fact that ranking is a complex task requiring many training samples to learn representations that can be used to assess the relevance, while relatively few data with strong labels are available.

Alternating between strong and weak data during training, i.e., NN$_{\text{W/S}^+}$ seems to bring little (but statistically significant) improvement. However, we can gain better results by the typical fine tuning strategies. Among the fine tuning experiments, updating all the parameters of the target network (or student), i.e., NN$_{\text{W}} \rightarrow$ NN$_{\text{S}}$, is the best fine tuning strategy. Updating only the parameters of the representation layer based on the strong labels, i.e., NN$_{\text{W}} \rightarrow$ NN$_{\text{S}}^{\text{Rep}}$, works better than updating only parameters of the supervision layer, i.e., NN$_{\text{W}} \rightarrow$ NN$_{\text{S}}^{\text{Sup}}$. This supports our designed choice of a shared embedding layer in CWS which gets updated on the set $\mathcal{D}_s$.

FWL is the best performing model, and CWS achieves 97% of the performance of FWL. The main advantage of CWS over FWL is that it is trained in a single stage process and needs to see all the samples in $\mathcal{D}_w$ (which is a reasonably large set) only one time, while FWL has three sequential stages during training and it needs to iterate two times over all the samples in $\mathcal{D}_w$. Also, employing a Gaussian Process as part of the model in FWL limits its scalability, while the components of CWS are all neural networks, and this eases the increase in the capacity of the model.

As an ablation study on CWS, we tried different training strategies and report the results in Table 5.3. As shown, CWS and CWS$_{\text{CT}}$ perform better than other

Table 5.3: Performance of variants of CWS on different datasets for document ranking task. Baselines are described in Table 5.1.

	Method	Robust04		ClueWeb	
		MAP	*nDCG@20*	*MAP*	*nDCG@20*
8	**CWS**	**0.3017**	**0.4511**	0.1363	**0.2444**
9	$\mathbf{CWS^{+}_{JT}}$	0.2786	0.4367	0.1310	0.2244
10	$\mathbf{CWS_{ST}}$	0.2716	0.4237	0.1320	0.2213
11	$\mathbf{CWS_{CT}}$	0.2961	0.4440	**0.1378**	0.2431
12	$\mathbf{CWS_{PT}}$	0.2784	0.4292	0.1314	0.2207

Table 5.4: Performance of FWL against some of the baselines on different datasets for document ranking task. Baselines are described in Table 5.1.

	Method	Robust04		ClueWeb	
		MAP	***nDCG@20***	*MAP*	***nDCG@20***
13	**FWL**	**0.3124**	**0.4607**	**0.1472**	**0.2453**
14	$\mathbf{NN_{W^{\omega}\to NN_S}}$	0.2899	0.4431	0.1320	0.2309
15	$\mathbf{FWL}_{unsuprep}$	0.2211	0.3700	0.0831	0.1964
16	$\mathbf{FWL}\backslash\Sigma$	0.2980	0.4516	0.1386	0.2340

strategies. CWS_{CT} is to let the confidence network to be trained separately, while still being able to enjoy shared learned information from the target network. Compared to CWS, CWS_{CT} is less efficient as we need two rounds of training on weakly labeled data.

While it seems reasonable to make use of strong labels for updating *all* parameters of the target network, $\text{CWS}^{+}_{\text{JT}}$ achieves no better results than CWS. We speculate that during training, the direction of the parameter optimization is profoundly affected by the type of supervision signal and while we control the magnitude of the gradients, we do not change their directions. Hence alternating between two sets with different label qualities (different supervision signal types, i.e., weak and strong) confuses the supervision layer of the target network.

In CWS_{ST}, the strong dataset, $\mathcal{D}_s$, is too small to train a high-quality confidence network without taking advantage of the vast amount of weakly annotated data in $\mathcal{D}_w$ to learn better representations, so CWS_{ST} is not able to improve the performance over NN_W significantly and also we noticed that this strategy leads to a slow convergence compared to the NN_W. Also, transferring learned information from the target network to the confidence network via progressive training, i.e., CWS_{PT}, performs no better than full sharing of the representation learning layer.

Table 5.4 presents the results of experiments we have done as ablation studies

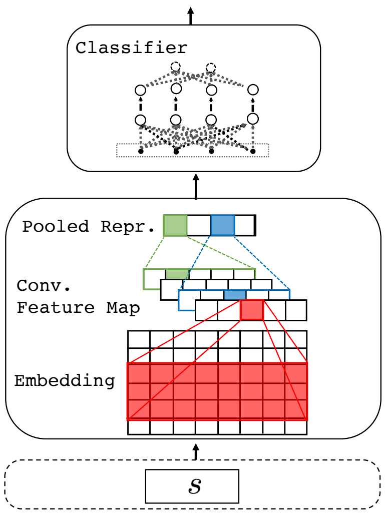


Figure 5.5: The sentiment classifier used as teacher in CWS and student in FWL.

on FWL.

Weighting the gradient updates from weak labels during pretraining and fine tuning the network with strong labels, i.e., $\text{NN}_{\text{W}^{\omega}\rightarrow\text{S}}$ seems to work quite well. Comparing the performance of $\text{FWL}_{unsuprep}$ to FWL indicates that, first of all, learning the representation of the input data downstream of the main task leads to better results compared to a task-independent unsupervised or self-supervised way. Also, the dramatic drop in performance compared to FWL, emphasizes the importance of the preretraining the student on weakly labeled data.

We can gain improvements by fine tuning the NN_{W} using labels generated by the teacher without considering their confidence score, i.e., $\text{FWL}\backslash\Sigma$. This means we just augment the fine tuning process by generating a fine tuning set using a teacher, which is better than $\mathcal{D}_s$ in terms of quantity and $\mathcal{D}_w$ in terms of quality. This baseline is equivalent to setting $\beta = 0$ in Equation 5.4. However, we see a big jump in performance when we use FWL to include the estimated label quality from the teacher, leading to the best overall results.

5.4.2 Sentiment Classification

In sentiment classification, the goal is to predict the sentiment (e.g., positive, negative, or neutral) of a sentence. Each training sample x consists of a sentence s and its sentiment label $\tilde{y}$.

The target network in CWS and the student in FWL

We use a convolutional model [160] as the target network in CWS and the student in FWL, which is similar to the state-of-the-art model for Twitter sentiment classification from Semeval 2015 and 2016 [85, 86, 250, 251].

The Representation Learning Layer The representation learning layer in this task consists of an embedding function $\varepsilon : \mathcal{V} \to \mathbb{R}^m$, where $\mathcal{V}$ denotes the vocabulary set and m is the dimension of the embedding.

This function maps the sentence to a matrix $S \in \mathbb{R}^{m\times|s|}$, where each column represents the embedding of a word at the corresponding position in the sentence. We initialize the embedding matrix with word2vec embeddings [195] pretrained on a collection of 50M tweets.

Matrix S is passed through a convolution layer. In this layer, a set of f filters is applied to a sliding window of length h over S to generate a feature map matrix O. Each feature map o_i for a given filter F is generated by $o_i = \sum_{k,j} S[i : i+h]_{k,j} F_{k,j}$, where $S[i : i+h]$ denotes the concatenation of word vectors from position i to $i+h$. The concatenation of all o_i produces a feature vector $o \in \mathbb{R}^{|s|-h+1}$. The vectors o are then aggregated over all f filters into a feature map matrix $O \in \mathbb{R}^{f\times(|s|-h+1)}$.

We also add a bias vector $b \in R^f$ to the result of a convolution. Each convolutional layer is followed by a non-linear activation function (we use ReLU [203]) which is applied element-wise. Afterward, the output is passed to a max pooling layer which operates on columns of the feature map matrix O returning the largest value: $pool(o_i) : \mathbb{R}^{1\times(|s|-h+1)} \to \mathbb{R}$.

The Supervision Layer. This layer is a simple fully connected feed-forward network with l hidden layers, followed by a softmax. We employ the weighted cross entropy loss:

$$\mathcal{L}_t = \sum_{i\in B_{\mathcal{D}_w}} \tilde{c}_i \sum_{k\in K} -\tilde{y}_i^k \log(\hat{y}_i^k), \tag{5.8}$$

where $B_{\mathcal{D}_w}$ is a batch of samples from $\mathcal{D}_w$, and $\tilde{c}_i$ is the confidence score of the weakly annotated sample i, and K is the set of classes. The general schema of the target network (or student) is illustrated in Figure 5.5.

The weak annotator

The weak annotator for the sentiment classification task is a simple lexicon-based method [122, 163]. We use SentiWordNet03 [19] to assign probabilities (positive, negative and neutral) for each token in set $\mathcal{D}_w$. We use a bag-of-words model for the sentence-level probabilities (i.e., just averaging the distributions of the terms), yielding a noisy label $\tilde{y}_i \in \mathbb{R}^{|K|}$, where $|K| = 3$ is the number of classes. We found empirically that using soft labels from the weak annotator works better than assigning a single hard label.

The confidence network in CWS

In this task, the confidence network is also a regressor and we use a simple fully connected feed-forward network. The target label c_j for the confidence network is calculated by using the mean absolute difference of the strong label and the weak label: $c_j = 1 - \frac{1}{|K|} \sum_{k \in K} |y_j^k - \tilde{y}_j^k|$, where y_j is the one-hot encoding of the sentence label over all classes.

The teacher in FWL

Similar to the ranking task, we use a Gaussian Process as the teacher in order to generate soft labels. We pass the mean of $\mathcal{GP}$ through the same function $g(\cdot)$ that is applied on the output of the student network, where the $g(\cdot)$ is softmax for the sentiment classification task. Here in this task, $h(\cdot)$ is an aggregation function that takes the variance over several dimensions and outputs a single measure of variance. As a reasonable choice, the aggregation function $h(\cdot)$ in the sentiment classification task (three classes) is the *mean* of variances over dimensions. In the teacher, linear combinations of different kernels are used. For the sentiment classification task, we use sparse variational GP for multiclass classification[7] [132] with the following kernel:

$$k(x_i, x_j) = k_{\text{RBF}}(x_i, x_j) + k_{\text{Linear}}(x_i, x_j) + k_{\text{White}}(x_i, x_j) \tag{5.9}$$

where

$$\begin{aligned} k_{\text{RBF}}(x_i, x_j) &= \exp\left(\frac{\|x_i - x_j\|^2}{2l^2}\right), \\ k_{\text{Linear}}(x_i, x_j) &= \sigma_0^2 + x_i.x_j, \\ k_{\text{White}}(x_i, x_j) &= \mathit{constant\_value}\ \forall x_1 = x_2, \text{ and 0 otherwise.} \end{aligned}$$

Similar to the ranking task, we set $l = 1$ as the length scale of the RBF kernel, $\sigma_0 = 0$ for the linear kernel, and the number of clusters to 30 in the clustered $\mathcal{GP}$ algorithm.

[7] *http://gpflow.readthedocs.io/en/latest/notebooks/multiclass.html*

Collections

We test our model on the Twitter message-level sentiment classification of SemEval-15 Task 10B [238]. Datasets of SemEval-15 subsume the test sets from previous editions of SemEval, i.e., SemEval-13 and SemEval-14. Each tweet was preprocessed so that URLs and usernames are masked.

Data with strong labels. We use train (9,728 tweets) and development (1,654 tweets) data from SemEval-13 for training and SemEval-13-test (3,813 tweets) for validation. To make your results comparable to the official runs on SemEval we us SemEval-14 (1,853 tweets) and SemEval-15 (2,390 tweets) as test sets [204, 238].

Data with weak labels. We use a large corpus containing 50M tweets collected during two months for both, training the word embeddings and creating the weakly annotated set $\mathcal{D}_w$ using the lexicon-based method explained in Section 5.4.2.

Experimental Setup

Similar to the document ranking task, we tune hyper-parameters for the target network in CWS (and student in the first step of FWL) with respect to the strong labels of the validation set using batched GP bandits with an expected improvement acquisition function [87] and keep the optimal parameters fixed for all the other experiments. The size and number of hidden layers for the classifier and s selected from $\{32, 64, 128\}$. We test the model with both, 1 and 2 convolutional layers. The number of convolutional feature maps and the filter width is selected from $\{200, 300\}$ and $\{3, 4, 5\}$, respectively. The initial learning rate and the dropout parameter are selected from $\{1E-3, 1E-5\}$ and $\{0.0, 0.2, 0.5\}$, respectively. We consider embedding sizes of $\{100, 200\}$ and the batch size in these experiments was set to 64. The ReLU [203] is used as a non-linear activation function in target network (and student). The Adam optimizer [161] is used for training, and dropout as a regularizer.

Results and Discussion

We report Macro-F1, the official SemEval metric, in Table 5.5.

Among all the baselines, FWL is the best performing approach. CWS also outperforms all the baselines.

For this task, since the amount of data with strong labels is larger than for the ranking task, the performance of $\mathrm{NN_S}$ is acceptable. Alternately sampling from weak and strong data, i.e., $\mathrm{NN_{W/S^+}}$ gives better results than either of learning from just weak or just strong labels. However, pretraining on weak labels and

Table 5.5: Performance of CWS and FWL as well as the main baseline methods, described in Table 5.1, for the sentiment classification task. $^{\blacktriangle i}$ indicates that the improvements with respect to baseline *i* are statistically significant at the 0.05 level using the paired two-tailed t-test with Bonferroni correction.

	Method	**SemEval-14**	**SemEval-15**
1	$\mathbf{WA_{Lexicon}}$	0.5141	0.4471
2	$\mathbf{NN_S}$	$0.6307^{\blacktriangle 1}$	$0.5811^{\blacktriangle 13}$
3	$\mathbf{NN_W}$	$0.6719^{\blacktriangle 12}$	$0.5606^{\blacktriangle 1}$
4	$\mathbf{NN_{W/S^+}}$	$0.7032^{\blacktriangle 12367}$	$0.6319^{\blacktriangle 12367}$
5	$\mathbf{NN_W \rightarrow NN_S}$	$0.7080^{\blacktriangle 12367}$	$0.6441^{\blacktriangle 12367}$
6	$\mathbf{NN_W \rightarrow NN_S^{Sup}}$	$0.6875^{\blacktriangle 123}$	$0.6193^{\blacktriangle 123}$
7	$\mathbf{NN_W \rightarrow NN_S^{Rep}}$	$0.6932^{\blacktriangle 123}$	$0.6102^{\blacktriangle 123}$
8	**CWS**	$0.7362^{\blacktriangle 1234567}$	$0.6626^{\blacktriangle 1234567}$
13	**FWL**	$\mathbf{0.7470}^{\blacktriangle 12345678}$	$\mathbf{0.6830}^{\blacktriangle 12345678}$
$*$	$\mathbf{SemEval^{Best}}$	0.7162 [240]	0.6618 [85]

Table 5.6: Performance of variants of CWS on different datasets for the sentiment classification task. Baselines are described in Table 5.1.

	Method	**SemEval-14**	**SemEval-15**
8	**CWS**	0.7362	0.6626
9	$\mathbf{CWS_{JT+}}$	0.7310	0.6551
10	$\mathbf{CWS_{ST}}$	0.7183	0.6501
11	$\mathbf{CWS_{CT}}$	**0.7363**	**0.6667**
12	$\mathbf{CWS_{PT}}$	0.7009	0.6118

then fine tuning both the supervision layer and the representation learning layer on strong labels, further improves the performance.

Besides the baselines, we also report the best performing systems that are also convolution-based models (Rouvier and Favre 240 on SemEval-14; Deriu et al. 85 on SemEval-15). Both CWS and FWL outperform these methods.

Similar to the ranking task, we have done an ablation study on CWS by trying different strategies for training CWS. The results of these experiments are presented in Table 5.6. CWS_{CT} archives the highest performance among all the training strategies, however, as we discussed in Section 5.4.1, it is not as efficient as CWS.

In sentiment classification, compared to the ranking task, it is easier to estimate the confidence score of samples concerning the amount of available supervised data. Therefore, CWS_{ST} improves the performance over NN_S in Table 5.5.

Table 5.7: Performance of FWL against some of the baselines on different datasets for document ranking task. Baselines are described in Table 5.1.

	Method	SemEval-14	SemEval-15
13	**FWL**	**0.7470**	**0.6830**
14	**$\mathrm{NN}_{\mathrm{W}^{\omega}} \to \mathrm{NN}_{\mathrm{S}}$**	0.7166	0.6603
15	**$\mathrm{FWL}_{unsuprep}$**	0.6588	0.6954
16	**$\mathrm{FWL}\backslash\Sigma$**	0.7202	0.6590

The results of a set of experiments we have done as ablation studies on FWL are presented in Table 5.7.

Having static weighting on the gradient updates, i.e., $\mathrm{NN}_{\mathrm{W}^{\omega}\to\mathrm{S}}$, leads to a performance that is better than simple fine tuning, i.e., $\mathrm{NN}_{\mathrm{W}} \to \mathrm{NN}_{\mathrm{S}}$ in Table 5.5. For this task, similar to the ranking task, learning the representation in an unsupervised task-independent fashion, i.e., $\mathrm{FWL}_{unsuprep}$, does not lead to good results compared to FWL. Similar to the ranking task, fine tuning NN_{W} based on labels generated by $\mathcal{GP}$ instead of data with strong labels, regardless of the confidence score, i.e., $\mathrm{FWL}\backslash\Sigma$ [293], works better than standard fine tuning.

5.5 Discussion and Analysis

In this section, we provide further analyses by investigating the learning pace in CWS and FWL, the bias-variance trade-off in FWL, the sensitivity of FWL to the quality of weak labels, and how modulating the learning rate in FWL can be different from the weighted sampling of training samples.

5.5.1 Faster Learning Pace in CWS

In CWS, controlling the contribution of the weak labels to updating the parameters of the model not only improves the performance, but also provides the network with more solid signals, which speeds up the learning process.

Figure 5.6 illustrates the training/validation loss for both networks compared to the loss of training the target network with weak supervision, along with their performance on test sets, with respect to different amounts of training data for the sentiment classification task we observed a similar pattern for the document ranking task.

As shown, in training, the loss of the target network in our model, i.e., $\mathcal{L}_t$ is higher than the loss of the network that is trained only on weakly supervised data, i.e., $\mathcal{L}_{\mathrm{NN}_{\mathrm{W}}}$. However, since these losses are calculated with respect to the weak labels (not true labels), having very low training loss can be an indication

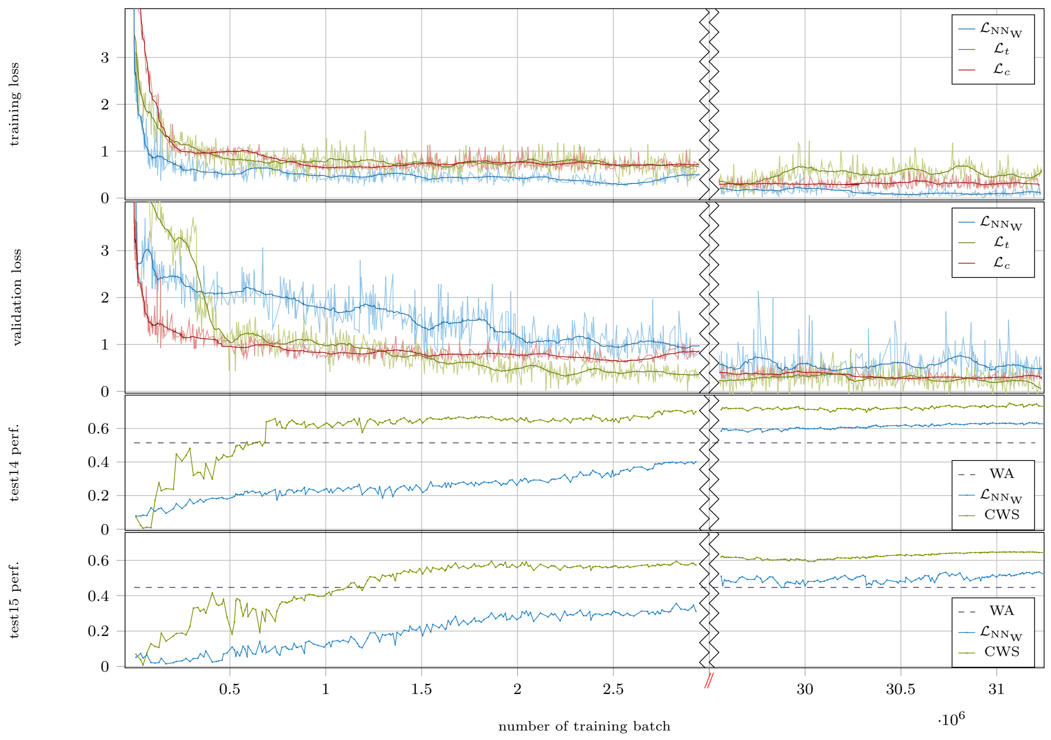


Figure 5.6: Loss of the target network ($\mathcal{L}_t$) and the confidence network ($\mathcal{L}_c$) compared to the loss of $\mathrm{NN_W}$ ($\mathcal{L}_{\mathrm{NN_W}}$) on training/validation set and performance of CWS, $\mathrm{NN_W}$, and weak annotator on test sets with respect to different amount of training data on sentiment classification.

of overfitting to the imperfection of weak labels. In other words, regardless of the general problem of lack of generalization due to overfitting, in the setup of learning from weak labels, predicting labels that are similar to train labels (very low training loss) is not necessarily a desirable incident.

In the validation set, however, $\mathcal{L}_t$ decreases faster than $\mathcal{L}_{\mathrm{NN_W}}$, which supports the fact that $\mathcal{L}_{\mathrm{NN_W}}$ overfits to the imperfection of weak labels, while our setup helps the target network to escape from this imperfection and do an excellent job on the validation set. In terms of the performance, compared to $\mathrm{NN_W}$, the performance of CWS on both test sets increases very quickly and CWS can pass the performance of the weak annotator by seeing much fewer instances annotated by the weak annotator.

5.5.2 A Good Teacher is Better than Many Observations

We look at the rate of learning for the student in FWL as the amount of training data is varied. This experiment is related to the connection of FWL with Vapnik's *learning using privileged information* (LUPI) [287, 288], thus, we first highlight this connection.

Connection with Vapnik's LUPI

FWL makes use of information from a small set of correctly labeled data to improve the performance of a semi-supervised learning algorithm. The main idea behind LUPI comes from the fact that humans learn much faster than machines. This can be due to the role that an *Intelligent Teacher* plays in human learning. In this framework, the training data is a collection of triplets

$$\{(x_1, y_1, x_1^*), \ldots, (x_n, y_n, x_n^*)\} \sim P^n(x, y, x^*), \tag{5.10}$$

where each (x_i, y_i) is a pair of feature-label and x_i^* is the additional information provided by an intelligent teacher to ease the learning process for the student. Additional information for each (x_i, y_i) is available only during training time and the learning machine must only rely on x_i at test time. The theory of LUPI studies how to leverage such a teaching signal x_i^* to outperform learning algorithms utilizing only the normal features x_i. For example, MRI brain images can be augmented with high-level medical or even psychological descriptions of Alzheimer's disease to build a classifier that predicts the probability of Alzheimer's disease from an MRI image at test time. It is known from statistical learning theory [289] that the following bound for test error is satisfied with probability $1 - \delta$:

$$R(f) \leq R_n(f) + O\left(\left(\frac{|\mathcal{F}|_{VC} - \log \sigma}{n}\right)^{\alpha}\right), \tag{5.11}$$

where $R_n(f)$ denotes the training error over n samples, $|\mathcal{F}|_{VC}$ is the VC dimension of the space of functions from which f is chosen, and $\alpha \in [0.5, 1]$. When the classes are not *separable*, $\alpha = 0.5$ i.e., the machine learns at a slow rate of $O(n^{-1/2})$. For easier problems where classes are *separable*, $\alpha = 1$ resulting in a learning rate of $O(n^{-1})$. The difference between these two cases is severe. The same error bound achieved for a separable problem with 10 thousand data points is only obtainable for a non-separable problem when 100 million data points are provided. This is prohibitive even when obtaining large datasets is not so costly.

The theory of LUPI shows that an intelligent teacher can reduce α resulting in a faster learning process for the student. In this chapter, we have proposed a *teacher-student* framework for semi-supervised learning. Similar to LUPI, in FWL a student is supposed to solve the main prediction task while an intelligent teacher provides additional information to improve its learning. In addition, we first train the student network so that it obtains initial knowledge of weakly labeled data and learns a good data representation. Then the teacher is trained on truly labeled data enjoying the representation learned by the student. This extends LUPI in a way that the teacher provides privileged information that is most useful for the current state of student's knowledge. FWL also extends

LUPI by introducing several teachers each of which is specialized to correct the student's knowledge related to a specific region of the data space.

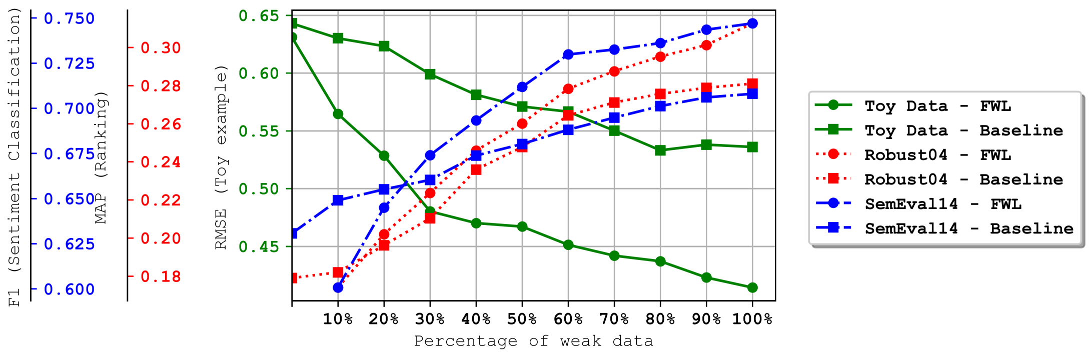
The theory of LUPI was first developed and proved for support vector machines by Vapnik as a method for knowledge transfer. Hinton introduced *dark knowledge* as a spiritually close idea in the context of neural networks [138]. He proposed to use a large network or an ensemble of networks for training and a smaller network at test time. It turned out that compressing knowledge of a large system into a smaller system can improve the generalization ability. It was shown in [180] that dark knowledge and LUPI can be unified under a single umbrella, called *generalized distillation*. The core idea of these models is *machines-teaching-machines*. As the name suggests, a machine is learning the knowledge embedded in another machine. In our case, the student is correcting its knowledge by receiving privileged information about label uncertainty from the teacher.

FWL extends the core idea of LUPI in the following directions:

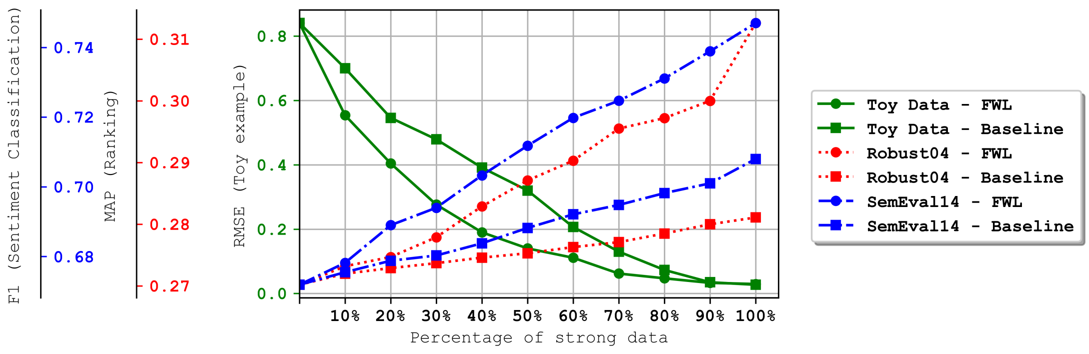
- Trainable teacher: It is often assumed that the teacher in LUPI framework has some additional true information. We show that when this extra information is not available, one can still use the LUPI setup and define an implicit teacher whose knowledge is learned from the true data. In this approach, the performance of the final student-teacher system depends on a clever answer to the following question: which information should be considered as the privileged knowledge of teacher.
- Bayesian teacher: The proposed teacher is Bayesian. It provides posterior uncertainty of the label of each sample.
- Mutual representation: We introduced module $\psi(\cdot)$ that learns a mutual embedding (representation) for both student and teacher. This is in particular interesting because it defines a two-way channel between teacher and student.
- Multiple teachers: We proposed a scalable method to introduce several teachers such that each teacher is specialized in a particular region of the data space.

Relevant to this, we performed two types of experiments on all tasks:

- In the first experiment, we use all the available strong data but consider different percentages of the entire weak dataset.
- In the second experiment, we fix the amount of weak data and provide the model with varying amounts of strong data.



(a) Models trained on different amount of weak data.



(b) Models trained on different amount of strong data.

Figure 5.7: Performance of FWL and the baseline model trained on different amount of data.

We use standard fine tuning with similar setups as for the baseline models. We fixed everything in the model and tried running the fine tuning step with different values for $\beta \in \{0.0, 0.1, 1.0, 2.0, 5.0\}$ in all the experiments. For the experiments on the toy problem in Section 5.5.1, the reported numbers are averaged over 10 trials. In the first experiment (i.e., Figure 5.7a), the size of sampled data data is: $|\mathcal{D}_s| = 50$ and $|\mathcal{D}_w| = 100$ (fixed) and for the second experiment (i.e., Figure 5.7a): $|\mathcal{D}_w| = 100$ and $|\mathcal{D}_s| = 10$ (fixed).

Figure 5.7 presents the results of these experiments. In general, for all tasks and both setups, the student learns faster when there is a teacher. One caveat is in the case where we have a very small amount of weak data. In this case, the student cannot learn a suitable representation in the first step, and hence the performance of FWL is rather low, as expected. It is highly unlikely that this situation occurs in reality as obtaining weakly labeled data is much easier than strong data.

The empirical observation of Figure 5.7 that our model learns more with less data can also be seen as evidence in support of another perspective to FWL, i.e. *learning using privileged information* [287].

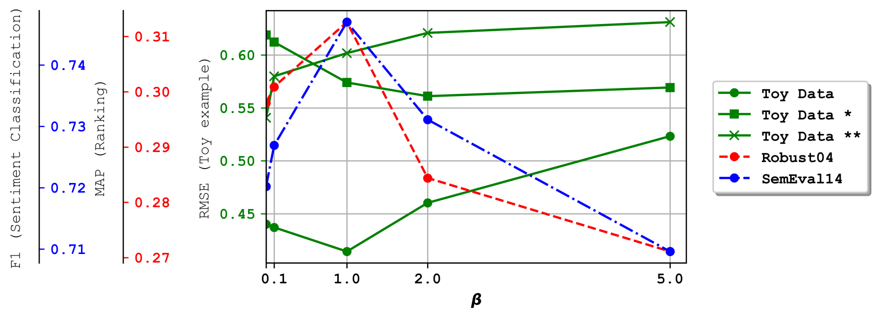


Figure 5.8: Effect of different values for β.

5.5.3 Handling the Bias-Variance Trade-off in FWL

As mentioned in Section 5.3.1, β is a hyperparameter that controls the contribution of weak and strong data to the training procedure. In order to investigate its influence, we fixed everything in the model and ran the fine tuning step with different values of $\beta \in \{0.0, 0.1, 1.0, 2.0, 5.0\}$ in all the experiments.

Figure 5.8 illustrates the performance on the ranking (on the Robust04 dataset) and sentiment classification tasks (on the SemEval14 dataset). For both sentiment classification and ranking, $\beta = 1$ gives the best results (higher scores are better). We also experimented on the toy problem with different values of β in three cases: 1) having 10 observations from the true function (same setup as Section 5.3.3), marked as "Toy Data" in the plot, 2) having only 5 observations from the true function, marked as "Toy Data *" in the plot, and 3) having $f(x) = x + 1$ as the weak function, which is an extremely bad approximator of the true function, marked as "Toy Data **" in the plot. For the "Toy Data" experiment, $\beta = 1$ turned out to be optimal (here, lower scores are better). However, for "Toy Data *," where we have an extremely small number of observations from the true function, setting β to a higher value acts as a regularizer by relying more on weak signals, and eventually leads to better generalization. On the other hand, for "Toy Data **," where the quality of the weak annotator is extremely low, lower values of β put more focus on the true observations. Therefore, β lets us control the bias-variance trade-off in these extreme cases.

We have also tested $\hat{c}_t = \eta_2(x_t) = {}^{\beta}/_{\mathrm{var}}[\Sigma(x_t)]$. The experiments showed that the exponential choice gives a better overall performance.

5.5.4 Sensitivity of FWL to the Quality of the Weak Annotator

Our proposed setup in FWL requires defining a so-called "weak annotator" to provide a source of weak supervision for unlabelled data. In Section 5.5.3 we discussed the role of the parameter β for controlling the bias-variance trade-off

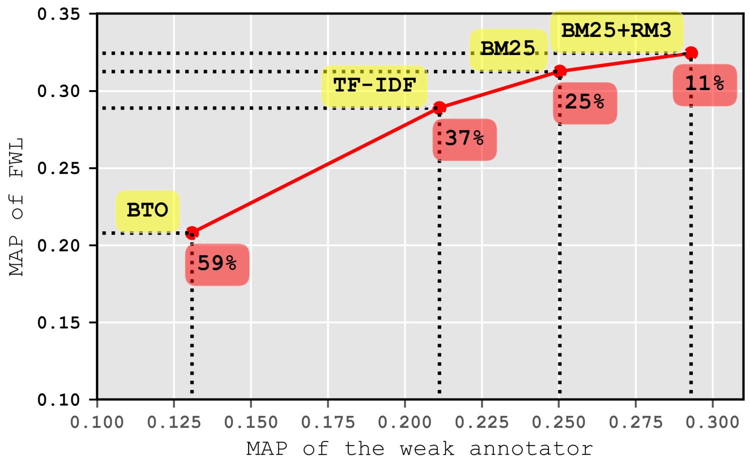


Figure 5.9: Performance of FWL versus performance of the corresponding weak annotator in the document ranking task, on the Robust04 dataset.

by trying two weak annotators for the toy problem. Now, in this section, we study how the quality of the weak annotator may affect the performance of the FWL, for the task of document ranking as a real-world problem.

To do so, besides BM25 [232], we use three other weak annotators: a vector space model [244] with a binary term occurrence (BTO) weighting schema and a vector space model with the TF-IDF weighting schema, which are both weaker than BM25, and BM25+RM3 [2] that uses RM3 as the pseudo-relevance feedback method on top of BM25, leading to better labels.

Figure 5.9 illustrates the performance of these four weak annotators in terms of their mean average precision (MAP) on the test data, versus the performance of FWL given the corresponding weak annotator. As expected, the performance of FWL depends on the quality of the employed weak annotator. The percentage of improvement of FWL over its corresponding weak annotator on the test data is also presented in Figure 5.9. As can be seen, the better the performance of the weak annotator is, the less the improvement of the FWL would be.

5.5.5 From Modifying the Learning Rate to Weighted Sampling

FWL provides confidence scores based on the certainty associated with each generated label $\bar{y}_t$, given sample $x_t \in \mathcal{D}_{sw}$. We can translate the confidence score as how likely including $(x_t, \bar{y}_t)$ in the training set for the student model improves the performance, and rather than using this score as the multiplicative factor in the learning rate, we can use it to bias the sampling procedure of mini-batches so that the frequency of training samples is proportional to the confidence score of their labels.

We design an experiment to try FWL with this setup (FWL$_s$), in which we keep

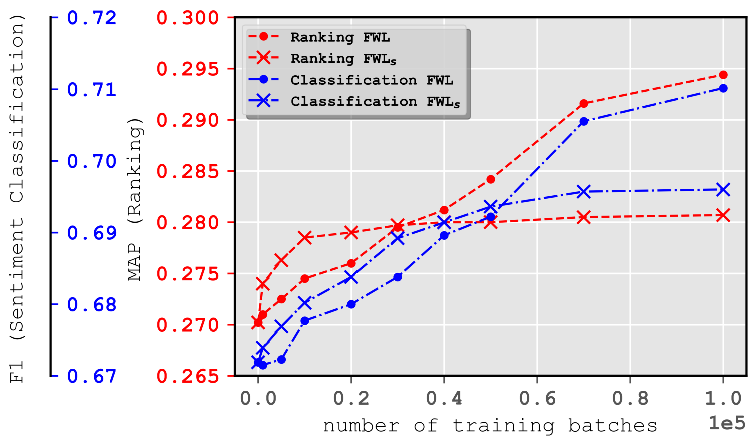


Figure 5.10: Performance of FWL and FWL$_s$ with respect to different batch of data for the task of document ranking (Robust04 dataset) and sentiment classification (SemEval14 dataset).

the architectures of the student and the teacher and the procedure of the first two steps of the FWL fixed, but we changed the step 3 as follows:

Given the soft dataset $\mathcal{D}_{sw}$, consisting of x_t, its label $\bar{y}_t$ and the associated confidence score generated by the teacher, we normalize the confidence scores over all training samples and set the normalized score of each sample as its probability to be sampled. Afterwards, we train the student model by mini-batches sampled from this set with respect to the probabilities associated with each sample, but without considering the original confidence scores in parameter updating. This means that the more confident the teacher is about the generated label for each sample, the more chance that the sample has to be seen by the student model. Figure 5.10 illustrates the performance of both FWL and FWL$_s$ trained on different amounts of data sampled from $\mathcal{D}_{sw}$, in the document ranking and sentiment classification tasks.

As can be seen, compared to FWL, the performance of FWL$_s$ increases rapidly in the beginning but it slows down afterward. We have looked into the sampling procedure and noticed that the confidence scores provided by the teacher form a rather skewed distribution and there is a strong bias in FWL$_s$ toward sampling from data points that are either in or close to the points in $\mathcal{D}_s$, as $\mathcal{GP}$ has less uncertainty around these points and the confidence scores are high. We observed that the performance of FWL$_s$ gets closer to the performance of FWL after many epochs, while FWL had already a log convergence. The skewness of the confidence distribution makes FWL$_s$ to have a tendency for more exploitation than exploration, however, FWL has more chance to explore the input space, while it controls the effect of updates on the parameters for samples based on their merit.

5.6 Related Work

In this section, we position the introduced CWS and FWL approaches relative to related work.

5.6.1 Learning from Imperfect Data

Learning from imperfect labels has been thoroughly studied in the literature [103]. The imperfect (weak) signal can come from non-expert crowd workers, be the output of other models that are weaker (for instance with low accuracy or coverage), biased, or models trained on data from different related domains. Among these forms, in the distant supervision setup, a heuristic labeling rule [85, 251] or function [84], which can be relying on a knowledge base [123, 196, 197] is employed to devise noisy labels.

Learning from weak data sometimes aims at encoding various forms of domain expertise or cheaper supervision from lay annotators. For instance, in structured learning, the label space is pretty complex and obtaining a training set with strong labels is extremely expensive, hence this class of problems leads to a wide range of works on learning from weak labels [239]. Indirect supervision is considered as a form of learning from weak labels that is employed in particular in the structured learning, in which a companion binary task is defined for which obtaining training data is easier [39, 224].

In the response-based supervision, the model receives feedback from interacting with an environment in a task, and converts this feedback into a supervision signal to update its parameters [51, 230, 239]. Constraint-based supervision is another form of weak supervision in which constraints that are represented as weak label distributions are taken as signals for updating the model parameters. For instance, physics-based constraints on the output [265] or output constraints on execution of logical forms [51].

In the proposed CWS and FWL, we can employ these approaches as the weak annotator to provide imperfect labels for the unlabeled data, however, a small amount of data with strong labels is also needed, which puts our model in the class of semi-supervised models.

Some noise cleaning methods have been proposed to remove or correct mislabeled samples [31]. There are some studies showing that weak or noisy labels can be leveraged by modifying the loss function [216, 217, 228, 284] or changing the update rule to avoid imperfections of noisy data [82, 83, 189].

One direction of research focuses on modeling the pattern of the noise or weakness in the labels. For instance, methods that use a generative model to correct weak labels such that a discriminative model can be trained more effectively [226, 229, 290]. Furthermore, methods that aim at capturing the pattern of the noise by inserting an extra layer [109] or a separate module tries to infer better labels by correcting the noisy labels and then use new labels for training [83, 266, 293]. Our proposed FWL can be categorized in this class as the teacher tries to infer better labels and provide certainty information which is incorporated as the update rule for the student model.

5.6.2 Semi- supervised Learning

In the semi-supervised setup, some ideas were developed to utilize weakly or even unlabeled data. For instance, the idea of self(incremental)-training [236], pseudo-labeling [137, 172], and co-training [26] have been introduced for augmenting the training set by unlabeled data with predicted labels. Some research has used the idea of self-supervised (or unsupervised) feature learning [94, 95, 205] to exploit different labelings that are freely available besides or within the data, and to use them as intrinsic signals to learn general-purpose features. These features, that are learned using a proxy task, are then used in a supervised task like object classification/detection or description matching.

As a common approach in semi-supervised learning, the unlabeled set can be used for learning the distribution of the data. In particular for neural networks, greedy layer-wise pre-training of weights using unlabeled data is followed by supervised fine tuning [86, 108, 138, 250, 251]. Other methods learn unsupervised encoding at multiple levels of the architecture jointly with a supervised signal [210, 303].

5.6.3 Sentiment Classification and Document Ranking

Sentiment classification is one of the key NLP tasks and SemEval provides standard benchmark datasets for this task [204, 237, 238]. Many models have been proposed based on neural networks for the sentiment classification task. For many datasets, the state-of-the-art results are from convolutional-based models that learn multiple word vector representations [160]. In our work, we adapt a CNN based architecture, which is proposed to be trained with the help of weak (distance) supervising [85, 250, 251] and has achieved best results on some SemEval datasets [86]. Document Ranking is also the core task of IR and some recent studies have applied neural networks on this task. Two main groups of models are those that learn representations for query and documents, independently, and then use a matching function [144, 201, 253], or models that try to capture interactions between query and document from the beginning [84, 118, 182, 308]. Here, we adapt one of the best rankers among all the previously proposed neural rankers that can be trained with weak supervision [79, 84, 314].

5.7 Conclusion

Training neural networks using large amounts of weakly annotated data is an attractive approach in scenarios where an adequate amount of data with true labels is not available, a situation which often arises in practice. In this chapter, to address **RQ-2.2**: "*Given a large set of weakly annotated samples and a small set of samples with high-quality labels, how can we best leverage the capacity of information in these sets to train a neural network?*," we introduced two semi-supervised learning approaches in the presence of weakly labeled data: Learning from Controlled Weak Supervision (CWS) and fidelity-weighted learning (FWL).

CWS is a meta-learning approach that we proposed to address **RQ-2.2.1**. It unifies learning to estimate the confidence score of weak annotations and training neural networks to learn a target task with controlled weak supervision, i.e., using weak labels to updating the parameters but taking their estimated confidence scores into account. This helps to alleviate updates from instances with unreliable labels that may harm the performance.

FWL is a student-teacher framework that we proposed to address **RQ-2.2.2**. In FWL the student network is in charge of learning a target task given a vast amount of samples with weak labels associated with fidelity scores that are generated by the teacher network. In FWL, we pretrain the student network on weak data to learn an initial task-dependent data representation, which we pass to the teacher along with the strong data. The teacher then learns to predict the strong data, but crucially, *based on the student's learned representation*. This then allows the teacher to generate new labeled training data from unlabeled data as well as fidelity scores for each sample in the data. Using samples in the new dataset, we update the parameters of the student network taking the fidelity scores into account to modulate the learning rate.

We applied both CWS and FWL to document ranking and sentiment classification, and empirically verified that they improve over well-performing alternative semi-supervised methods and speed up the training process. We observed that the common approach of pre-training and fine tuning is not as effective. We showed that we can learn to explicitly model label qualities as a useful property of the data, which is not immediately available, and directly exploit this property in the learning process to address the fundamental challenge of training data quality-quantity trade-off.

In Part II to address **RQ-2**: "*How to design learning algorithms that can learn from weakly annotated samples, while generalizing over the imperfection in their labels?*," we explored different ideas for developing models that are capable of learning from weakly annotated data. We started with exploring how different architectural choices and different objective functions can be employed for learning with pseudo-labels that are programmatically generated to augment training data. Then we studied how to meta-learn the quality of labels and how to incorporate them in the learning process. In the next part, we study how we can employ inductive biases as modeling assumptions to design models that are not only effective, but also data-efficient.

Part III

Injecting Inductive Biases for Data Efficiency

Generalization is at the heart of many aspects of human cognition. It underlies our ability to learn language, form and use categories, and learn about causal relationships, while in many cases we are presented only with limited observation. The great ability of generalization in human cognition sets a standard to which artificial intelligence and machine learning research aspires.

This immediately raises questions like "How can human learn so much about the world from such limited evidence?" and "What makes human so good at generalization?" The importance of generalization in cognitive science partly derives from its close relationship to inductive inference. We can define generalization as forming predictions about future events based on examples from the past, which emphasizes its relationship to the classic problem of induction [145]. Having this connection between generalization and induction in mind, we can cast the question of "how to produce good generalizations?" to the question of "what makes a good inductive learner?"

To answer this question from a human cognition perspective, as humans develop, they become capable of exploring "more sophisticated hypotheses" about the structure of their environment [146], which allows them to better infer their reality status and become better inductive learners. However, humans sometimes have to make decisions without information from their senses or testimony from others, no matter how rich their hypothesis space is. In these cases, their internal dispositions, i.e., "inductive biases", provides a source of knowledge that influences their decisions in the absence of experience, explicit sensory or testimonial proof [116, 261].

In the context of learning algorithms, given the data d, the learner aims at identifying the hypothesis h from a set of hypotheses H that results in the highest generalization accuracy. As with human cognition, algorithms that are able to explore "richer hypothesis spaces" have the potential of being better inductive learners.[8] However, in many cases, the considered hypotheses are not directly defined by the observed data and to choose among the many hypotheses that are equally granted by the data, the learner has to inject some preferences for those hypotheses that are called "inductive biases".[9]

Inductive biases are factors that lead a learner to favor one hypothesis over another that are independent of the observed data. When two hypotheses fit the data equally well, inductive biases are the only basis for deciding between them and making it possible to generalize beyond the observed data [198].

This brings us to the discussion of the *bias-variance trade-off*. We can decompose the expected generalization error into two parts: the bias of a learning algorithm,

[8]The idea of expansion of the hypothesis space by adding more layers and non-linearity to neural networks to make it possible to overcome the constraint of linear separability.

[9]Biases can be both in inductive and in deductive systems. Systems that learn concepts from labeled training instances employ *inductive bias*.

and its variance [106]. The transition from one source of error to another is known as the *bias-variance trade-off*, and much of the research in designing learning algorithms is about trying to find the sweet spot between bias and variance for a given problem.

The bias-variance trade-off suggests that the generalization of a learning algorithm depends on the problems at hand and different factors involved in learning, like the amount of available data. If the learner is provided with only small amounts of data, then the variance is the real concern and richer hypothesis spaces may hurt the generalization because they increase variance. To increase the chance of generalization in these cases is to inject inductive biases with respect to the problem at hand. However, if the learner is provided with large amounts of data and needs to be able to solve a variety of problems, then variance is less of an issue and the bias can be the dominant source of error, thus the learner needs richer hypothesis spaces to be flexible enough to accommodate the different solutions, as this reduces bias.

For neural networks, the inductive biases inherent in their architecture is perhaps the most important factor determining how quickly they train and how well they generalize beyond the data they observed during training. A well-known example is the translation invariance assumption in convolutional neural networks [171] for vision tasks based on a certain symmetry in the data, which considers that a given feature, of any complexity, can appear anywhere in the image. Other examples include neural networks that encode rotational invariance [54] or permutation invariance [173, 319]. Having inductive biases gets even more crucial when we need data *efficient models* that are able to *generalize beyond observed training data* and can *learn complex tasks* like tasks that need reasoning or inferring an underlying structure from the data.

In Part III of this thesis, we address the following research question:

> **RQ-3** *How can inductive biases help to improve the generalization and data efficiency of learning algorithms?*

In this part, we focus on some of the sequence processing neural networks and study the role of inductive biases on the generalization and data efficiency of these models. Among sequence processing neural networks, recurrent neural networks (RNNs) have long been the de facto choice for sequence modeling tasks. The most important facet of RNNs is the *recurrence* which lets the model updates its internal state in a loop given the input at each time step. The recurrence is simply using information from the previous state which in turn uses the previous state so on and so forth. In other words, RNNs implement the "re-occurrence" of referring back to all previous internal states. This adds a "recurrent inductive bias" to the model that may be crucial for several algorithmic and language understanding tasks [75, 281].

However, the recurrence in time dictates the inherently sequential computation which makes RNNs slow to train. Feed-forward and convolutional architectures have recently been shown to achieve superior results on some sequence modeling tasks such as machine translation [155, 292], with the added advantage that they concurrently process all inputs in the sequence, leading to easy parallelization and faster training times. Despite these successes, however, popular feed-forward sequence models like the Transformer [292] fail to generalize in many simple tasks, e.g., copying strings or even simple logical inference when the string or formula lengths exceed those observed at training time, while recurrent models handle these tasks with ease because of the inductive recurrent bias.

In Chapter 6, we study how lack of recurrent inductive bias in Transformer can lead to the failure of the model on complex reasoning tasks with limited data, algorithmic tasks where length generalization over training samples is needed, and structured language understanding tasks, and address the following research question:

> **RQ-3.1** *How can we improve the generalization and data efficiency of self-attentive feed-forward sequence models by injecting a recurrent inductive bias?*

We introduced Universal Transformer [75], a self-attentive concurrent-recurrent sequence model, which is an extension of the Transformer model [292]. The Universal Transformer introduces recurrence in depth by repeatedly modifying a series of vector representations for each position of the sequence in parallel, by combining information from different positions using self-attention and applying a recurrent transition function across all time steps. In the simplest form, Universal Transformer with a fixed number of iterations is almost equivalent to a multi-layer Transformer with tied parameters across all its layers. By sharing weights, we can save massively on the number of parameters that we are training and fewer parameters means learn faster with fewer data points.

We show that the elegant idea of introducing recurrence in depth enables the Universal Transformer to extrapolate from training data much better on a range of algorithmic and language understanding tasks [72, 75]. Besides the recurrence in depth, we propose a simple inductive bias for UTs that lets the model generalize well to input lengths that are not observed during training. We also add a dynamic per-position halting mechanism and find that it improves accuracy on several tasks. It is noteworthy that in contrast to the standard Transformer, under certain assumptions UTs can be shown to be *Turing-complete*.

6 Recurrent Inductive Bias for Transformers

Inductive biases are effective means for encoding modeling assumptions and improving data efficiency. For many sequence modeling tasks, recurrent inductive bias is required for generalizing beyond the observed data, but recent self-attentive feed-forward sequence models trade recurrence for parallelizability. We can, however, introduce a form of recurrent inductive bias to these models to improve their generalization while keeping parallelization in the computations.

6.1 Introduction

A human child achieves the needed complex linguistic knowledge within a short time, with very limited observation. This cannot be explained easily and relying on the *poverty of stimulus* [48] argument, a powerful task-specific bias is required for learning to understand and generate language. In the context of learning algorithms, it is well-known that without a certain complexity in the learning bias, the training data is insufficient to permit learning a model that generalizes properly to the full range of unseen instances for a specific task [198]. In other words, to have machines that can generalize, *data-driven* learning, which merely relies on previous experience has to come together with

This chapter is based on [72, 75].

an *innately primed* learning to facilitate that some of the knowledge is encoded in the model, in the form of biases, either strong or weak.

Different neural network based models have been proposed for sequence modeling and employed for language understanding and generation tasks. Among them, convolutional and fully-intentional feed-forward architectures like the Transformer [291] have recently emerged as viable alternatives to recurrent neural networks (RNNs) for a range of sequence modeling tasks, notably machine translation [105, 291]. These parallel-in-time architectures address a significant shortcoming of RNNs, namely their inherently sequential computation which prevents parallelization across elements of the input sequence, whilst still addressing the vanishing gradients problem as the sequence length gets longer [140].

The Transformer model relies entirely on a self-attention mechanism [177, 214] to compute a series of context-informed vector-space representations of the symbols in its input and output, which are then used to predict distributions over subsequent symbols as the model predicts the output sequence symbol-by-symbol. Not only is this mechanism straightforward to parallelize, but as each symbol's representation is also directly informed by all other symbols' representations, this results in an effectively global receptive field across the whole sequence. This stands in contrast to, e.g., convolutional architectures which typically only have a limited receptive field.

The Transformer with its fixed stack of distinct layers foregoes RNNs' inductive bias towards learning iterative or recursive transformations. Our experiments indicate that this inductive bias may be crucial for several algorithmic and language understanding tasks of varying complexity: in contrast to models such as the Neural Turing Machine [115], the Neural GPU [154] or Stack RNNs [151], the Transformer does not generalize well to input lengths not encountered during training.

In this chapter, we address the following research question:

> **RQ-3.1** *How can we improve the generalization and data efficiency of self-attentive feed-forward sequence models by injecting a recurrent inductive bias?*

We introduce the *Universal Transformer (UT)*, a parallel-in-time recurrent self-attentive sequence model that can be cast as a generalization of the Transformer model, yielding increased theoretical capabilities and improved results on a wide range of challenging sequence-to-sequence tasks. UTs combine the parallelizability and global receptive field of feed-forward sequence models like the Transformer with the recurrent inductive bias of RNNs, which seems to be better suited to a range of algorithmic and natural language understanding sequence-to-sequence problems. As the name implies, and in contrast to the

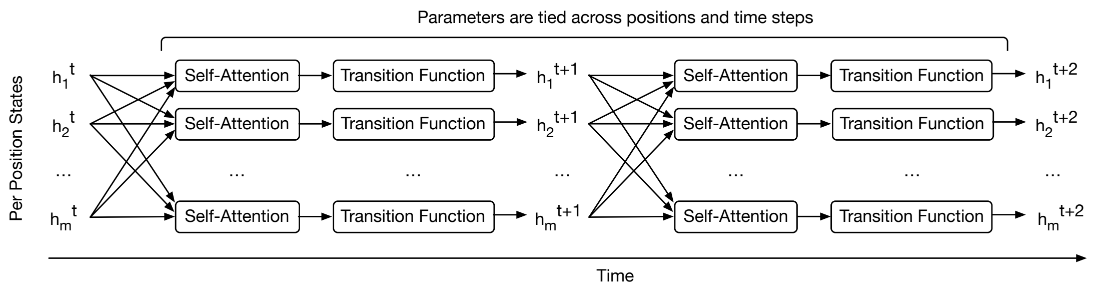


Figure 6.1: The Universal Transformer repeatedly refines a series of vector representations for each position of the sequence in parallel, by combining information from different positions using self-attention (see Eq. 6.3) and applying a recurrent transition function (see Eq. 6.5) across all time steps $1 \leq t \leq T$. We show this process over two recurrent time-steps. Arrows denote dependencies between operations. Initially, h^0 is initialized with the embedding for each symbol in the sequence. h_i^t represents the representation for input symbol $1 \leq i \leq m$ at recurrent time-step t. With dynamic halting, T is dynamically determined for each position (Section 6.2.1).

standard Transformer, under certain assumptions UTs can be shown to be Turing-complete (or "computationally universal", as shown in Section 6.3).

In each recurrent step, the Universal Transformer iteratively refines its representations for all symbols in the sequence in parallel using a self-attention mechanism [177, 214], followed by a transformation (shared across all positions and time-steps) consisting of a depth-wise separable convolution [44, 152] or a position-wise fully-connected layer (see Figure 6.1). We also add a dynamic per-position halting mechanism [114], allowing the model to choose the required number of refinement steps *for each symbol* dynamically, and show for the first time that such a conditional computation mechanism can in fact improve accuracy on several smaller, structured algorithmic and linguistic inference tasks (although it marginally degraded results on MT).

Our strong experimental results show that UTs outperform Transformers and LSTMs across a wide range of tasks. The added recurrence yields improved results in machine translation where UTs outperform the standard Transformer. In experiments on several algorithmic tasks and the bAbI language understanding task, UTs also consistently and significantly improve over LSTMs and the standard Transformer. Furthermore, on the challenging LAMBADA text understanding data set UTs with dynamic halting achieve new state of the art results.

6.1.1 Detailed Research Questions

We break down our main research question in this chapter into two concrete research questions:

> **RQ-3.1.1** *How do Universal Transformers combine the recurrent inductive bias of RNNs with the parallelizability and global receptive field of the Transformer?*
>
> **RQ-3.1.2** *How effective are Universal Transformers at complex reasoning tasks with limited data, at algorithmic tasks that need generalization over observed training samples, and at real-world language understanding tasks?*

In the following sections, we will address these research questions.

6.2 The Universal Transformer

Here, in this section, we focus on the following research question:

> **RQ-3.1.1** *How do Universal Transformers combine the recurrent inductive bias of RNNs with the parallelizability and global receptive field of the Transformer?*

The Universal Transformer (UT; see Figure 6.2) is based on the popular encoder-decoder architecture commonly used in most neural sequence-to-sequence models [43, 270, 291]. Both the encoder and decoder of the UT operate by applying a recurrent neural network to the representations of each of the positions of the input and output sequence, respectively. However, in contrast to most applications of recurrent neural networks to sequential data, the UT does not recur over positions in the sequence, but over consecutive revisions of the vector representations of each position (i.e., over "depth"). In other words, the UT is not computationally bound by the number of symbols in the sequence, but only by the number of revisions made to each symbol's representation.

In each recurrent time-step, the representation of every position is concurrently (in parallel) revised in two sub-steps: first, using a self-attention mechanism to exchange information across all positions in the sequence, thereby generating a vector representation for each position that is informed by the representations of all other positions at the previous time-step. Then, by applying a transition function (shared across position and time) to the outputs of the self-attention mechanism, independently at each position.

As the recurrent transition function can be applied any number of times, this implies that UTs can have variable depth (number of per-symbol processing steps). Crucially, this is in contrast to most popular neural sequence models, including the Transformer [291] or deep RNNs, which have constant depth as

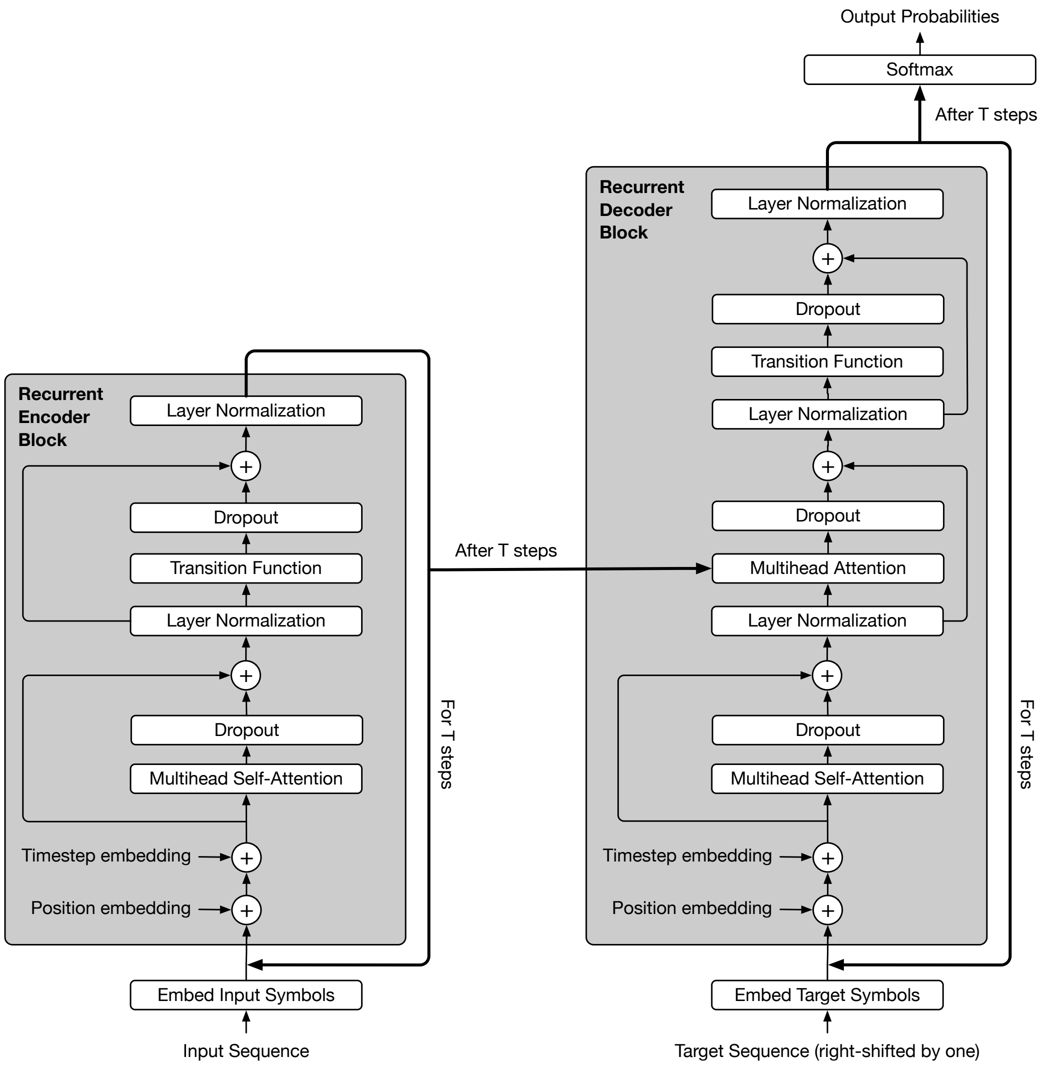


Figure 6.2: The recurrent blocks of the Universal Transformer encoder and decoder.

a result of applying a *fixed stack* of layers. We now describe the encoder and decoder in more detail (See Figure 6.2 for the schema of the complete model.).

ENCODER:

Given an input sequence of length m, we start with a matrix whose rows are initialized as the d-dimensional embeddings of the symbols at each position of the sequence $H^0 \in \mathbb{R}^{m \times d}$. The UT then iteratively computes representations H^t at step t for all m positions in parallel by applying the multi-headed dot-product self-attention mechanism from [291], followed by a recurrent transition function. We also add residual connections around each of these function blocks and apply dropout and layer normalization [18, 264].

Before describing the details of the first block of the universal transformer, i.e., the multi-head attention, we explain the attention mechanism in general as the main building block of many of state-of-the-art models. An attention mechanism is usually based on a memory-query paradigm. In this setup, there exist a memory M that contains a collection of items, each representing information from a source modality (like the hidden state of the encoder), and a query q vector that contains information from a target modality (like the hidden state of the decoder).[1] Each item in the memory is associated with a key and value (k_i, and v_i), where the key is used to compute the probability that indicates how well the query matches the item:

$$a_i = \frac{exp(f_{\text{att}}(q, k_i)}{\sum_{j=1}^{|} M|exp(f_{\text{att}}(q, k_j)}, \tag{6.1}$$

where the f_{att} can be the dot product function [184] or a multilayer perceptron [20]. Then, given the attention distributions over all items in the memory which is calculated by querying the memory with query q, we can output a value that is a sum over all the values in the memory weighted by their probabilities, which can be fed to other parts of the model for further calculation.

In the Universal Transformer, similar to the Transformer, we use the scaled dot-product attention which combines queries Q, keys K and values V as follows

$$\textsc{Attention}(Q, K, V) = \textsc{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) V, \tag{6.2}$$

where d is the number of columns of Q, K and V. We use the multi-head version with k heads, as introduced in [291]:

$$\textsc{MultiHeadSelfAttention}(H^t) = \textsc{Concat}(\text{head}_1, ..., \text{head}_\text{k})W^O \tag{6.3}$$

$$\text{where head}_\text{i} = \textsc{Attention}(H^t W_i^Q, H^t W_i^K, H^t W_i^V), \tag{6.4}$$

and we map the state H^t to queries, keys and values with affine projections using learned parameter matrices $W^Q \in \mathbb{R}^{d \times d/k}$, $W^K \in \mathbb{R}^{d \times d/k}$, $W^V \in \mathbb{R}^{d \times d/k}$ and $W^O \in \mathbb{R}^{d \times d}$.

At step t, the UT then computes revised representations $H^t \in \mathbb{R}^{m \times d}$ for all m input positions as follows:

$$H^t = \textsc{LayerNorm}(A^t + \textsc{Transition}(A^t)) \tag{6.5}$$

$$\text{where } A^t = \textsc{LayerNorm}((H^{t-1} + P^t) + \tag{6.6}$$

$$\textsc{MultiHeadSelfAttention}(H^{t-1} + P^t)), \tag{6.7}$$

[1]In self-attention, queries and memory are from the same modality.

where LAYERNORM() is defined in [18], and TRANSITION() and P^t are discussed below.

Depending on the task, we use one of two different transition functions: either a separable convolution [44] or a fully-connected neural network that consists of a single rectified-linear activation function between two affine transformations, applied position-wise, i.e., individually to each row of A^t.

$P^t \in \mathbb{R}^{m \times d}$ above are fixed, constant, two-dimensional (position, time) *coordinate embeddings*, obtained by computing the sinusoidal position embedding vectors as defined in [291] for the positions $1 \leq i \leq m$ and the time-step $1 \leq t \leq T$ separately for each vector-dimension $1 \leq j \leq d$, and summing:

$$P^t_{i,2j} = \sin(i/10000^{2j/d}) + \sin(t/10000^{2j/d}) \tag{6.8}$$

$$P^t_{i,2j+1} = \cos(i/10000^{2j/d}) + \cos(t/10000^{2j/d}). \tag{6.9}$$

After T steps (each updating all positions of the input sequence in parallel), the final output of the Universal Transformer encoder is a matrix of d-dimensional vector representations $H^T \in \mathbb{R}^{m \times d}$[2] for the m symbols of the input sequence.

DECODER:

The decoder shares the same basic recurrent structure of the encoder. However, after the self-attention function, the decoder additionally also attends to the final encoder representation H^T of each position in the input sequence using the same multihead dot-product attention function from Equation 6.3, but with queries Q obtained from projecting the *decoder* representations, and keys and values (K and V) obtained from projecting the *encoder* representations (this process is akin to standard attention [20]).

Like the Transformer model, the UT is autoregressive [113]. Trained using teacher-forcing, at generation time it produces its output one symbol at a time, with the decoder consuming the previously produced output positions. During training, the decoder input is the target output, shifted to the right by one position. The decoder self-attention distributions are further masked so that the model can only attend to positions to the left of any predicted symbol. Finally, the per-symbol target distributions are obtained by applying an affine transformation $O \in \mathbb{R}^{d \times V}$ from the final decoder state to the output vocabulary size V, followed by a softmax which yields an $(m \times V)$-dimensional output matrix normalized over its rows:

$$p\left(y_{pos}|y_{[1:pos-1]}, H^T\right) = \text{SOFTMAX}(OH^T) \tag{6.10}$$

[2]Note that T in H^T denotes time-step T and not the transpose operation.

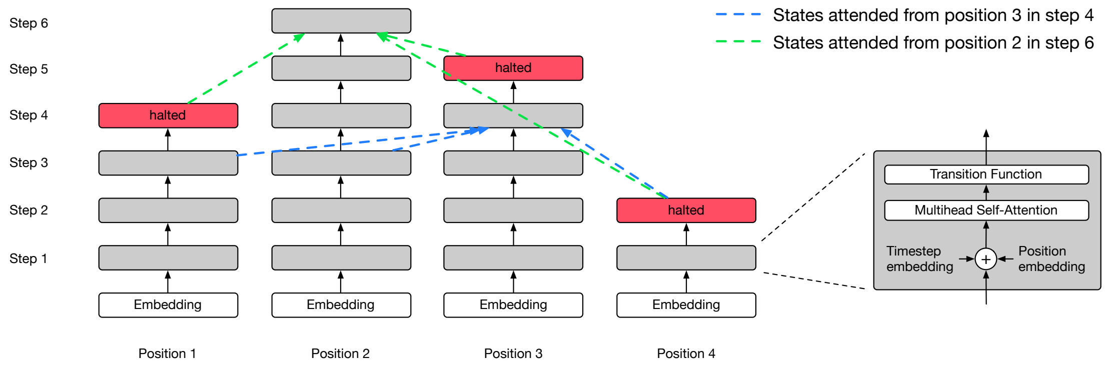


Figure 6.3: An unrolled visualization of Universal Transformer with dynamic halting. It illustrates different numbers of recurrent revisions per position (best viewed in color).

To generate from the model, the encoder is run once for the conditioning input sequence. Then the decoder is run repeatedly, consuming all already-generated symbols, while generating one additional distribution over the vocabulary for the symbol at the next output position per iteration. We then typically sample or select the highest probability symbol as the next symbol.

6.2.1 Adaptive Computation by Dynamic Halting

In sequence processing systems, certain symbols (e.g., some words or phonemes) are usually more ambiguous than others. It is therefore reasonable to allocate more processing resources to these more ambiguous symbols. Adaptive Computation Time (ACT) [114] is a mechanism for dynamically modulating the number of computational steps needed to process each input symbol (called the “ponder time”) in standard recurrent neural networks based on a scalar *halting probability* predicted by the model at each step.

Inspired by the interpretation of Universal Transformers as applying self-attentive RNNs in parallel to all positions in the sequence, we also add a dynamic ACT halting mechanism to each position, i.e., to each per-symbol self-attentive RNN. Once the per-symbol recurrent block halts, its state is simply copied to the next step until all blocks halt, or we reach a maximum number of steps. The final output of the encoder is then the final layer of representations produced in this way. Figure 6.3 illustrates a Universal Transformer encoder with T, number of revisions, dynamically determined for each position.

We implement dynamic halting based on ACT [114]. In each step of the UT with dynamic halting, we are given the halting probabilities, remainders, number of updates up to that point, and the previous state (all initialized as zeros), as well as a scalar threshold between 0 and 1 (a hyper-parameter). We then compute the new state for each position and calculate the new per-position halting probabilities based on the state for each position.[3] The UT then decides to halt for some positions that crossed the threshold, and updates the state of other positions until the model halts for all positions or reaches a predefined maximum number of steps. The following is a simplified TensorFlow [98] code that shows the details for the adapted ACT mechanism.

```python
# While−loop stops when this predicate is FALSE
# i.e., all ((probability < threshold) & (counter < max_steps)) are
    false
def should_continue(u0, u1, halting_probability, u2, n_updates, u3):
return tf.reduce_any(
            tf.logical_and(
                tf.less(halting_probability, threshold),
                tf.less(n_updates, max_steps)))
# Do while loop iterations until predicate above is false
(_, _, _, remainder, n_updates, new_state) = tf.while_loop(
    should_continue, ut_with_dynamic_halting, (state,
    step, halting_probability, remainders, n_updates, previous_state
    ))
```

Listing 6.1: UT with dynamic halting.

The following shows the computations in each step:

```python
def ut_with_dynamic_halting(state, step, halting_probability,
                             remainders, n_updates, previous_state):
    # Calculate the probabilities based on the state
    p = common_layers.dense(state, 1, activation=tf.nn.sigmoid,
        use_bias=True)
    # Mask for inputs which have not halted yet
    still_running = tf.cast(
        tf.less(halting_probability,1.0), tf.float32)
    # Mask of inputs which halted at this step
    new_halted = tf.cast(
        tf.greater(halting_probability + p * still_running,
    threshold),
            tf.float32) * still_running
    # Mask of inputs which haven't halted, and didn't halt this step
    still_running = tf.cast(
        tf.less_equal(halting_probability + p * still_running,
            threshold), tf.float32) * still_running
    # Add the halting probability for this step to the halting
    # probabilities for those inputs which haven't halted yet
    halting_probability += p * still_running
    # Compute remainders for the inputs which halted at this step
    remainders += new_halted * (1 − halting_probability)
```

[3]The current implementation of adaptive computation time does not allow for a fully end-to-end backpropable gradient of the proposed training loss. However, the discontinuity of the cost function might not imply that meaningful learning is not possible and in fact, the experiments in the original paper [114] as well as here in Universal Transformer with adaptive halting suggest it works fine.

```
    # Add the remainders to those inputs which halted at this step
    halting_probability += new_halted * remainders
    # Increment n_updates for all inputs which are still running
    n_updates += still_running + new_halted
    # Compute the weight to be applied to the new state and output:
    #   0 when the input has already halted,
    #   p when the input hasn't halted yet,
    #   the remainders when it halted this step.
    update_weights = tf.expand_dims(p * still_running +
                                    new_halted * remainders, -1)
    # Apply transformation to the state
    transformed_state = transition_function(self_attention(state))
    # Interpolate transformed and previous states for non-halted
   inputs
    new_state = ((transformed_state * update_weights) +
                 (previous_state * (1 - update_weights)))
    step += 1
    return (transformed_state, step, halting_probability,
            remainders, n_updates, new_state)
```

Listing 6.2: Computations in each step of the UT with dynamic halting.

6.3 Universality and Relation to other Models

When running for a fixed number of steps, the Universal Transformer is equivalent to a multi-layer Transformer with tied parameters across all its layers. This is partly similar to the Recursive Transformer, which ties the weights of its self-attention layers across depth [117].[4] However, as the per-symbol recurrent transition functions can be applied any number of times, another and possibly more informative way of characterizing the UT is as a block of parallel RNNs (one for each symbol, with shared parameters) evolving per-symbol hidden states concurrently, generated at each step by attending to the sequence of hidden states at the previous step. In this way, it is related to architectures such as the Neural GPU [154] and the Neural Turing Machine [115].

UTs thereby retain the attractive computational efficiency of the original feed-forward Transformer model, but with the added recurrent inductive bias of RNNs. Furthermore, using a dynamic halting mechanism, UTs can choose the number of processing steps based on the input data.

The connection between the Universal Transformer and other sequence models is apparent from the architecture: if we limited the recurrent steps to one, it would be a Transformer. But it is more interesting to consider the relationship between the Universal Transformer and RNNs and other networks where

[4]Note that in UT both the self-attention and transition weights are tied across layers.

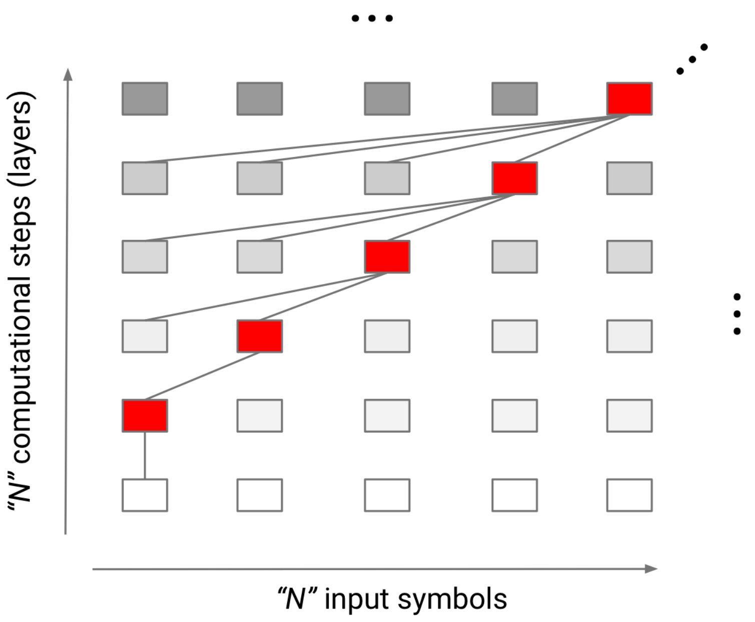
recurrence happens over the time dimension. Superficially these models may seem closely related since they are recurrent as well. But there is a crucial difference: time-recurrent models like RNNs cannot access memory in the recurrent steps. This makes them computationally more similar to automata, since the only memory available in the recurrent part is a fixed-size state vector. UTs, on the other hand, can attend to the whole previous layer, allowing it to access memory in the recurrent step.

Given sufficient memory the Universal Transformer is computationally universal – i.e., it belongs to the class of models that can be used to simulate any Turing machine, thereby addressing a shortcoming of the standard Transformer model. In addition to being theoretically appealing, our results show that this added expressivity also leads to improved accuracy on several challenging sequence modeling tasks. This closes the gap between practical sequence models competitive on large-scale tasks such as machine translation, and computationally universal models such as the Neural Turing Machine or the Neural GPU [115, 154], which can be trained using gradient descent to perform algorithmic tasks.

To show this, we can reduce a Neural GPU to a Universal Transformer. Ignoring the decoder and parameterizing the self-attention module, i.e., self-attention with the residual connection, to be the identity function, we assume the transition function to be a convolution. If we now set the total number of recurrent steps T to be equal to the input length, we obtain exactly a Neural GPU. Note that the last step is where the Universal Transformer crucially differs from the vanilla Transformer whose depth cannot scale dynamically with the size of the input.

A similar relationship exists between the Universal Transformer and the Neural Turing Machine, whose single read/write operations per step can be expressed by the global, parallel representation revisions of the Universal Transformer. In contrast to these models, however, which only perform well on algorithmic tasks, the Universal Transformer also achieves competitive results on realistic natural language tasks such as LAMBADA and machine translation.

Another related model architecture is that of end-to-end Memory Networks [267]. In contrast to end-to-end memory networks, however, the Universal Transformer uses memory corresponding to states aligned to individual positions of its inputs or outputs. Furthermore, the Universal Transformer follows the encoder-decoder configuration and achieves competitive performance in large-scale sequence-to-sequence tasks.



6.3.1 On the Computational Power of UT vs Transformer

With respect to their computational power, the key difference between the Transformer and the Universal Transformer lies in the number of sequential steps of computation (i.e., in depth). While a standard Transformer executes a total number of operations that scales with the input size, the number of sequential operations is constant, independent of the input size and determined solely by the number of layers. Assuming finite precision, this property implies that the standard Transformer cannot be computationally universal. When choosing a number of steps as a function of the input length, however, the Universal Transformer does not suffer from this limitation. Note that this holds independently of whether or not adaptive computation time is employed but does assume a non-constant, even if possibly deterministic, number of steps. Varying the number of steps dynamically after training is enabled by sharing weights across sequential computation steps in the Universal Transformer.

An intuitive example are functions whose execution requires the sequential processing of each input element. In this case, for any given choice of depth T, one can construct an input sequence of length $N > T$ that cannot be processed correctly by a standard Transformer. With an appropriate, input-length dependent choice of sequential steps, however, a Universal Transformer, RNNs or Neural GPUs can execute such a function.

6.4 Universal Transformer for Sequence Modeling

In this section, we address the following research question:

> **RQ-3.1.2** *How effective are Universal Transformers at complex reasoning tasks with limited data, at algorithmic tasks that need generalization over observed training samples, and at real-world language understanding tasks?*

We evaluate Universal Transformers on a range of algorithmic and language understanding tasks and discuss the results. In the tasks that are chosen, we include some with a limited number of training samples, or some with an incomplete training set (lack of converge), but also tasks that do not suffer severely from imperfect supervision, to evaluate the performance of our model in both situations.

In all the experiments, we made sure that the number of trainable parameters in the UT and the baselines are similar to have fair comparisons in terms of capacity of the models.

6.4.1 bAbI Question-Answering

The bAbi question answering dataset [302] consists of 20 different synthetic tasks.[5] The aim is that each task tests a unique aspect of language understanding and reasoning, including the ability to reason from supporting facts in a story, answer true/false type questions, count, understand negation and indefinite knowledge, understand coreferences, time reasoning, positional and size reasoning, path-finding, and understanding motivations (to see examples for each of these tasks, please refer to Table 1 and 2 in Section 3 of [302]).

There are two versions of the dataset, one with 1k training samples and the other with 10k samples. It is important for a model to be data-efficient to achieve good results using only the 1k training samples. Moreover, the original idea is that a single model should be evaluated across all the tasks (not tuning per task), which is the *train joint* setup in Tables 6.1 and 6.2.

Solving all the bAbI tasks by training a model on the 1k training dataset is challenging as some of the tasks are rather complex and with only 1k samples for each task, its hard for most of models to generalize well. So data efficiency should be a key property to be considered here. We tried a standard Transformer and observed that it does not achieve good results on bAbI tasks.[6] However,

[5] *https://research.fb.com/downloads/babi*

[6] We experimented with different hyper-parameters and different network sizes, but it always overfits.

Table 6.1: Average error and number of failed tasks ($>$ 5% error) out of 20 (in parentheses; lower is better in both cases) on the bAbI dataset under the different training/evaluation setups. We indicate state-of-the-art where available for each, or '-' otherwise.

Model	**10K samples**		**1K samples**	
	train single	*train joint*	*train single*	*train joint*
Previous best results:				
QRNet [249]	0.3 (0/20)	-	-	-
Sparse DNC [223]	-	2.9 (1/20)	-	-
GA+MAGE [91]	-	-	8.7 (5/20)	-
MemN2N [267]	-	-	-	12.4 (11/20)
Our results:				
Transformer [291]	15.2 (10/20)	22.1 (12/20)	21.8 (5/20)	26.8 (14/20)
Universal Transformer (this work)	0.23 (0/20)	0.47 (0/20)	5.31 (5/20)	8.50 (8/20)
UT w/ dynamic halting (this work)	**0.21 (0/20)**	**0.29 (0/20)**	**4.55 (3/20)**	**7.78 (5/20)**

we have designed a model based on the Universal Transformer which achieves state-of-the-art results bAbI task.

To encode the input, similar to [131], we first encode each fact in the story by applying a learned multiplicative positional mask to each word's embedding, and summing up all embeddings. We embed the question in the same way, and then feed the (Universal) Transformer with these embeddings of the facts and questions.

As originally proposed, models can either be trained on each task separately ("train single") or jointly on all tasks ("train joint"). Table 6.1 summarizes our results. We conducted 10 runs with different initializations and picked the best model based on performance on the validation set, similar to previous work. Both the UT and UT with dynamic halting achieve state-of-the-art results on all tasks in terms of average error and number of failed tasks,[7] in both the 10K and 1K training regime. Tables 6.2 presents the results of best and average results of 10 runs breakdown by task.

To understand the working of the model better, we analyzed both the attention distributions and the average ACT ponder times for this task.

First, we observe that the attention distributions start out very uniform, but get progressively sharper in later steps around the correct supporting facts that are required to answer each question, which is indeed very similar to how humans would solve the task.

Second, with dynamic halting we observe that the average ponder time (i.e., depth of the per-symbol recurrent processing chain) over all positions in all samples in the test data for tasks requiring three supporting facts is higher

[7]Defined as $>$ 5% error.

Table 6.2: Detailed results on the bAbI question answering tasks.

Best seed run for each task (out of 10 runs)				
Task id	**10K**		**1K**	
	train single	*train joint*	*train single*	*train joint*
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.5
3	0.4	1.2	3.7	5.4
4	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.5
6	0.0	0.0	0.0	0.5
7	0.0	0.0	0.0	3.2
8	0.0	0.0	0.0	1.6
9	0.0	0.0	0.0	0.2
10	0.0	0.0	0.0	0.4
11	0.0	0.0	0.0	0.1
12	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.6
14	0.0	0.0	0.0	3.8
15	0.0	0.0	0.0	5.9
16	0.4	1.2	5.8	15.4
17	0.6	0.2	32.0	42.9
18	0.0	0.0	0.0	4.1
19	2.8	3.1	47.1	68.2
20	0.0	0.0	2.4	2.4
avg err	0.21	0.29	4.55	7.78
failed	0	0	3	5

Average (±var) over all seeds (for 10 runs)				
Task id	**10K**		**1K**	
	train single	*train joint*	*train single*	*train joint*
1	0.0 ±0.0	0.0 ±0.0	0.2 ±0.3	0.1 ±0.2
2	0.2 ±0.4	1.7 ±2.6	3.2 ±4.1	4.3 ±11.6
3	1.8 ±1.8	4.6 ±7.3	9.1 ±12.7	14.3 ±18.1
4	0.1 ±0.1	0.2 ±0.1	0.3 ±0.3	0.4 ±0.6
5	0.2 ±0.3	0.8 ±0.5	1.1 ±1.3	4.3 ±5.6
6	0.1 ±0.2	0.1 ±0.2	1.2 ±2.1	0.8 ±0.4
7	0.3 ±0.5	1.1 ±1.5	0.0 ±0.0	4.1 ±2.9
8	0.3 ±0.2	0.5 ±1.1	0.1 ±0.2	3.9 ±4.2
9	0.0 ±0.0	0.0 ±0.0	0.1 ±0.1	0.3 ±0.3
10	0.1 ±0.2	0.5 ±0.4	0.7 ±0.8	1.3 ±1.6
11	0.0 ±0.0	0.1 ±0.1	0.4 ±0.8	0.3 ±0.9
12	0.2 ±0.1	0.4 ±0.4	0.6 ±0.9	0.3 ±0.4
13	0.2 ±0.5	0.3 ±0.4	0.8 ±0.9	1.1 ±0.9
14	1.8 ±2.6	1.3 ±1.6	0.1 ±0.2	4.7 ±5.2
15	2.1 ±3.4	1.6 ±2.8	0.3 ±0.5	10.3 ±8.6
16	1.9 ±2.2	0.9 ±1.3	9.1 ±8.1	34.1 ±22.8
17	1.6 ±0.8	1.4 ±3.4	43.7 ±18.6	51.1 ±12.9
18	0.3 ±0.4	0.7 ±1.4	2.3 ±3.6	12.8 ±9.0
19	3.4 ±4.0	6.1 ±7.3	50.2 ±8.4	73.1 ±23.9
20	0.0 ±0.0	0.0 ±0.0	3.2 ±2.5	2.6 ±2.8
avg	0.73 ±0.89	1.12 ±1.62	6.34 ±3.32	11.21 ±6.65

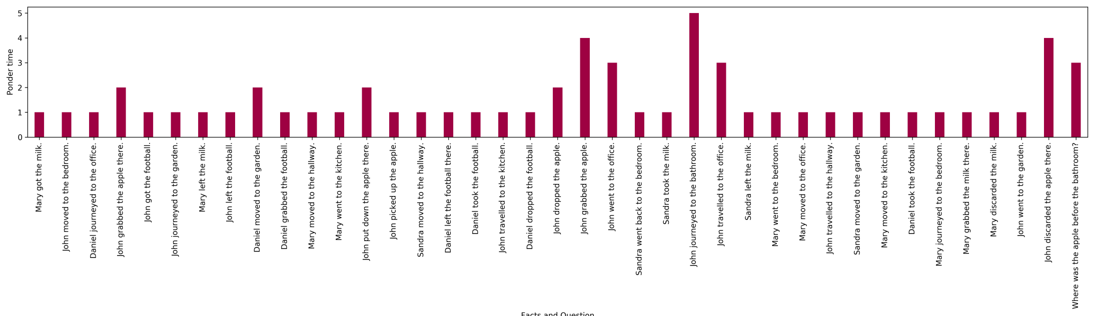


Figure 6.4: Ponder time of UT with dynamic halting (trained on 1k data with joint training) for encoding facts in a story and question in a bAbI task#3 requiring three supporting facts.

(3.8±2.2) than for tasks requiring only two (3.1±1.1), which is in turn higher than for tasks requiring only one supporting fact (2.3±0.8). This indicates that the model adjusts the number of processing steps with the number of supporting facts required to answer the questions.

Finally, we observe that the histogram of ponder times at different positions is more uniform in tasks requiring only one supporting fact compared to two and three, and likewise for tasks requiring two compared to three. Especially for tasks requiring three supporting facts, many positions halt at step 1 or 2 already and only a few get transformed for more steps (see, for example, Figure 6.4). This is particularly interesting as the length of stories is indeed much higher in this setting, with more irrelevant facts which the model seems to successfully learn to ignore in this way.

Similar to dynamic memory networks [166], there is an iterative attention process in UTs that allows the model to condition its attention over memory on the result of previous iterations.

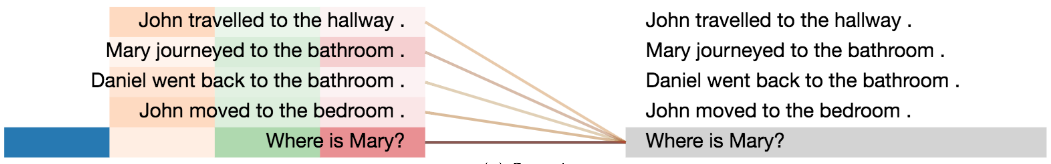
Figures 6.5, 6.6, 6.7, and 6.8 present visualizations of the self-attention distributions on bAbI tasks for some examples from Task 1, 2, and 3. The visualization of attention weights is over different time steps based on different heads over all the facts in the story and a question.

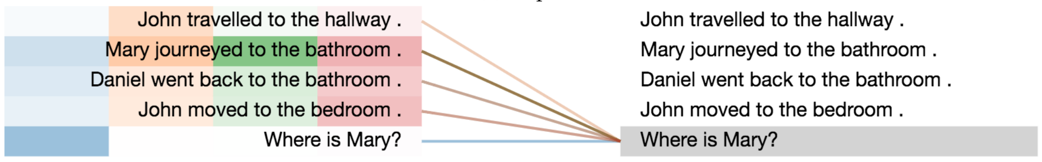
In all these examples, we visualize the attention distributions when transforming the question representation (right hand side) in the encoder. Different color bars on the left side indicate attention weights based on different heads (4 heads in total) over facts and the question.

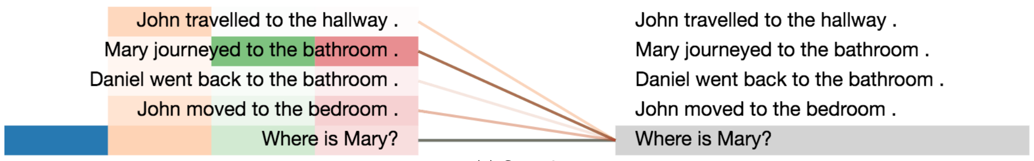
Although attention distributions are not necessarily explanations of the process or the outcome [147], these examples illustrate that there is a notion of temporal states in UT, where the model updates its states (memory) in each step based on the output of previous steps, and this chain of updates can also be viewed as steps in a multi-hop reasoning process.

The results of the Universal Transformer on bAbI tasks is a proxy of ability of the model to learn to do some basic reasoning, while being data efficient.

```
An example from tasks 1:   (requiring one supportive fact to solve)
Story:
                           John travelled to the hallway.
                           Mary journeyed to the bathroom.
                           Daniel went back to the bathroom.
                           John moved to the bedroom
Question:
                           Where is Mary?
Model's output:
                           bathroom
```


John travelled to the hallway .
Mary journeyed to the bathroom .
Daniel went back to the bathroom .
John moved to the bedroom .
Where is Mary?
John travelled to the hallway .
Mary journeyed to the bathroom .
Daniel went back to the bathroom .
John moved to the bedroom .
Where is Mary?
(a) Step 1


John travelled to the hallway .
Mary journeyed to the bathroom .
Daniel went back to the bathroom .
John moved to the bedroom .
Where is Mary?
John travelled to the hallway .
Mary journeyed to the bathroom .
Daniel went back to the bathroom .
John moved to the bedroom .
Where is Mary?
(b) Step 2


John travelled to the hallway .
Mary journeyed to the bathroom .
Daniel went back to the bathroom .
John moved to the bedroom .
Where is Mary?
John travelled to the hallway .
Mary journeyed to the bathroom .
Daniel went back to the bathroom .
John moved to the bedroom .
Where is Mary?
(c) Step 3

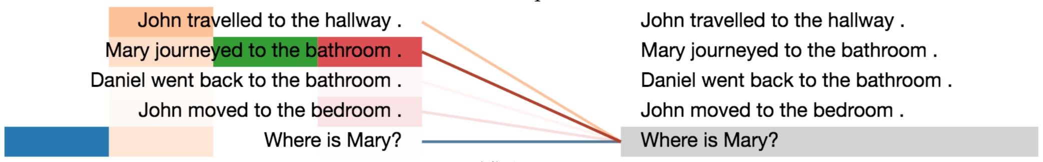
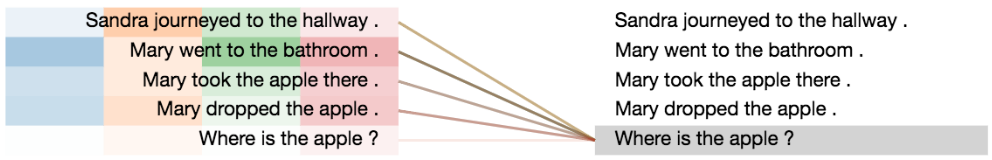
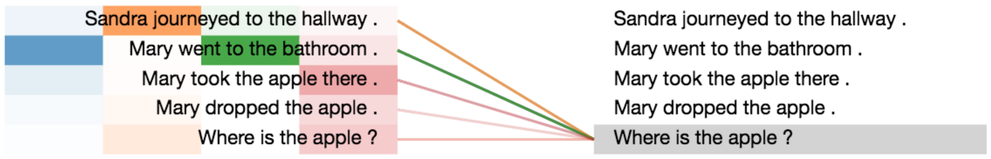

John travelled to the hallway .
Mary journeyed to the bathroom .
Daniel went back to the bathroom .
John moved to the bedroom .
Where is Mary?
John travelled to the hallway .
Mary journeyed to the bathroom .
Daniel went back to the bathroom .
John moved to the bedroom .
Where is Mary?
(d) Step 4

Figure 6.5: Visualization of the attention distributions, when encoding the question: *"Where is Mary?"*.

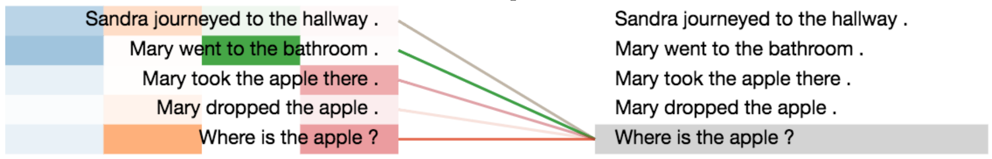
```
An example from tasks 2:   (requiring two supportive facts to solve)
Story:
                           Sandra journeyed to the hallway.
                           Mary went to the bathroom.
                           Mary took the apple there.
                           Mary dropped the apple.
Question:
                           Where is the apple?
Model's output:
                           bathroom
```


Sandra journeyed to the hallway .
Mary went to the bathroom .
Mary took the apple there .
Mary dropped the apple .
Where is the apple ?

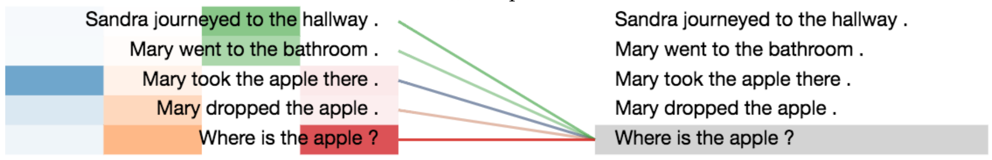
(a) Step 1


Sandra journeyed to the hallway .
Mary went to the bathroom .
Mary took the apple there .
Mary dropped the apple .
Where is the apple ?

(b) Step 2


Sandra journeyed to the hallway .
Mary went to the bathroom .
Mary took the apple there .
Mary dropped the apple .
Where is the apple ?

(c) Step 3


Sandra journeyed to the hallway .
Mary went to the bathroom .
Mary took the apple there .
Mary dropped the apple .
Where is the apple ?

(d) Step 4

Figure 6.6: Visualization of the attention distributions, when encoding the question: *"Where is the apple?"*.

```
An example from tasks 2:   (requiring two supportive facts to solve)

Story:
                           John went to the hallway.
                           John went back to the bathroom.
                           John grabbed the milk there.
                           Sandra went back to the office.
                           Sandra journeyed to the kitchen.
                           Sandra got the apple there.
                           Sandra dropped the apple there.
                           John dropped the milk.
Question:
                           Where is the milk?
Model's output:
                           bathroom
```

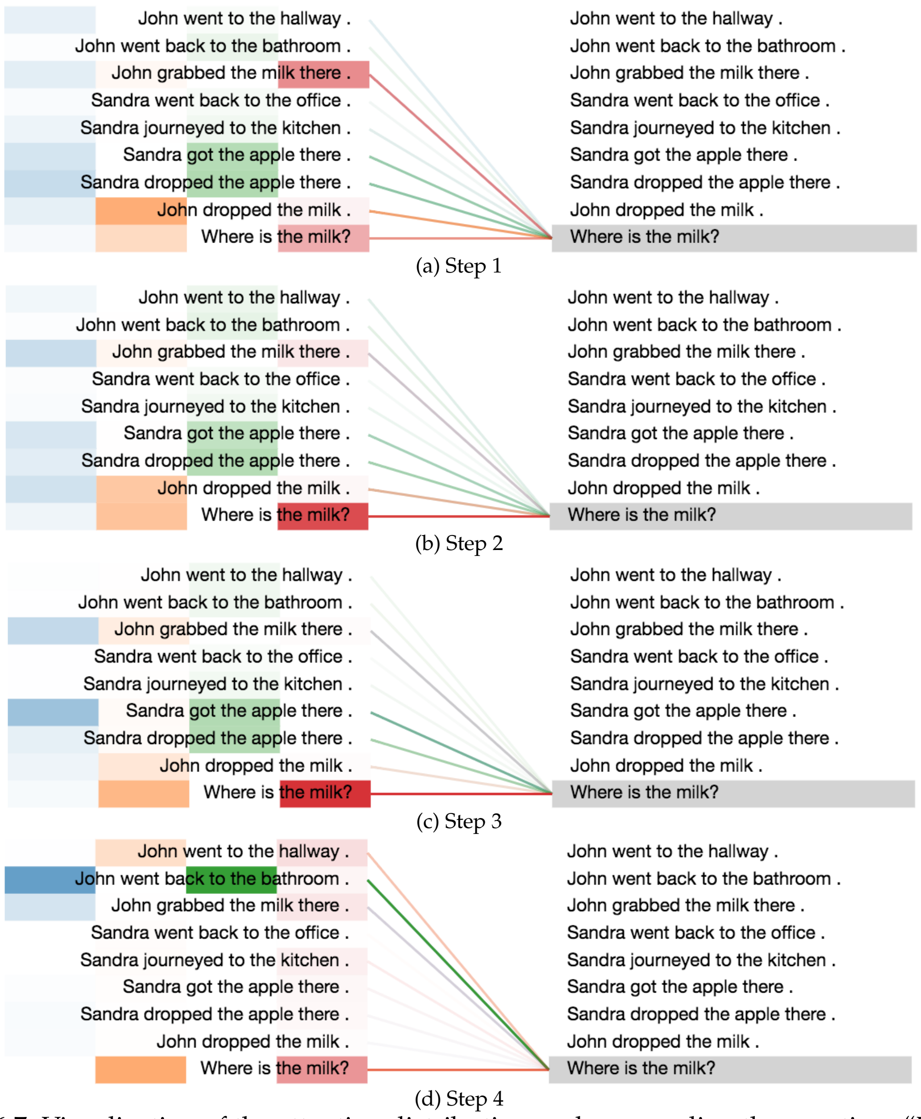
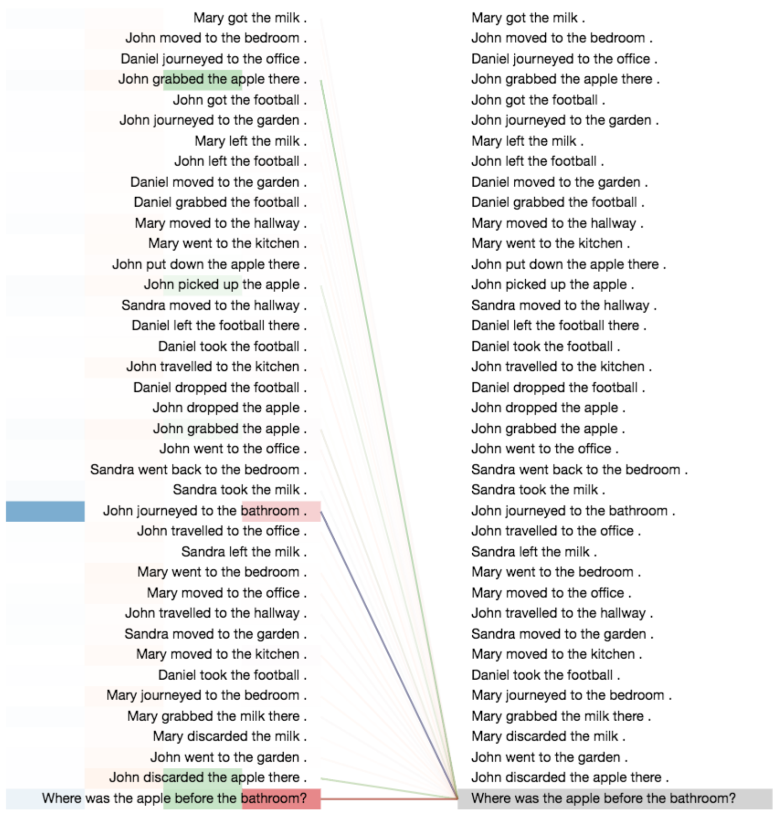
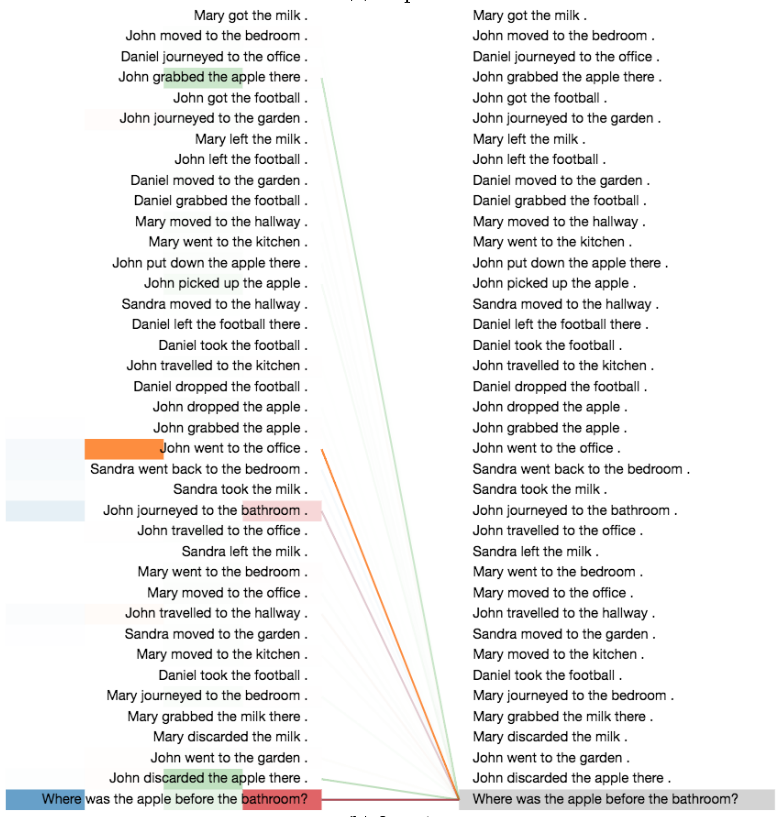


Figure 6.7: Visualization of the attention distributions, when encoding the question: *"Where is the milk?"*.

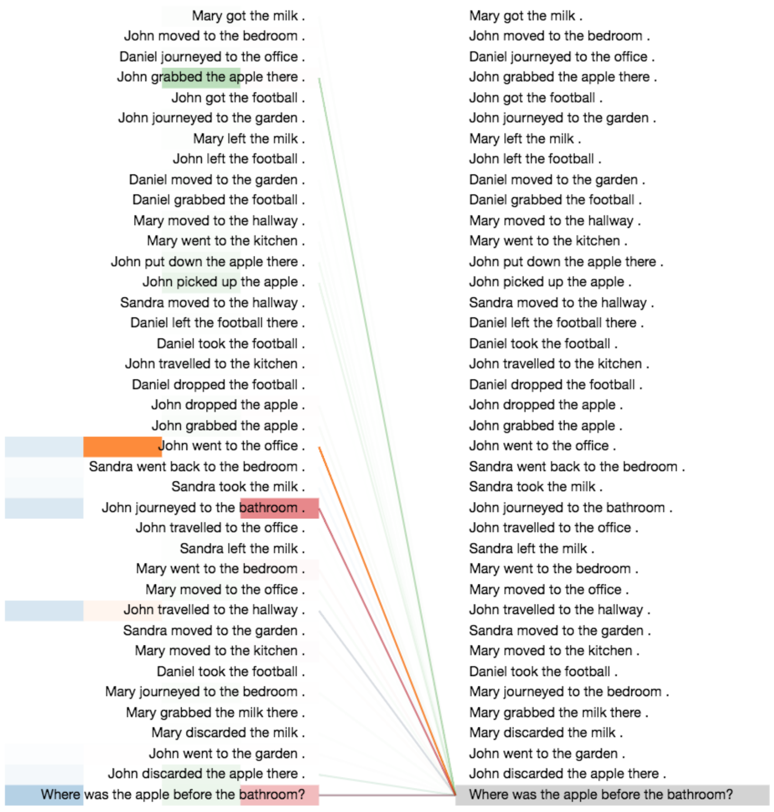
```
An example from tasks 3:   (requiring three supportive facts to solve)

Story:
Mary got the milk.
                          John moved to the bedroom.
                          Daniel journeyed to the office.
                          John grabbed the apple there.
                          John got the football.
                          John journeyed to the garden.
                          Mary left the milk.
                          John left the football.
                          Daniel moved to the garden.
                          Daniel grabbed the football.
                          Mary moved to the hallway.
                          Mary went to the kitchen.
                          John put down the apple there.
                          John picked up the apple.
                          Sandra moved to the hallway.
                          Daniel left the football there.
                          Daniel took the football.
                          John travelled to the kitchen.
                          Daniel dropped the football.
                          John dropped the apple.
                          John grabbed the apple.
                          John went to the office.
                          Sandra went back to the bedroom.
                          Sandra took the milk.
                          John journeyed to the bathroom.
                          John travelled to the office.
                          Sandra left the milk.
                          Mary went to the bedroom.
                          Mary moved to the office.
                          John travelled to the hallway.
                          Sandra moved to the garden.
                          Mary moved to the kitchen.
                          Daniel took the football.
                          Mary journeyed to the bedroom.
                          Mary grabbed the milk there.
                          Mary discarded the milk.
                          John went to the garden.
                          John discarded the apple there.

Question:
                          Where was the apple before the bathroom?
Model's output:
                          office
```


Mary got the milk .
John moved to the bedroom .
Daniel journeyed to the office .
John grabbed the apple there .
John got the football .
John journeyed to the garden .
Mary left the milk .
John left the football .
Daniel moved to the garden .
Daniel grabbed the football .
Mary moved to the hallway .
Mary went to the kitchen .
John put down the apple there .
John picked up the apple .
Sandra moved to the hallway .
Daniel left the football there .
Daniel took the football .
John travelled to the kitchen .
Daniel dropped the football .
John dropped the apple .
John grabbed the apple .
John went to the office .
Sandra went back to the bedroom .
Sandra took the milk .
John journeyed to the bathroom .
John travelled to the office .
Sandra left the milk .
Mary went to the bedroom .
Mary moved to the office .
John travelled to the hallway .
Sandra moved to the garden .
Mary moved to the kitchen .
Daniel took the football .
Mary journeyed to the bedroom .
Mary grabbed the milk there .
Mary discarded the milk .
John went to the garden .
John discarded the apple there .
Where was the apple before the bathroom?

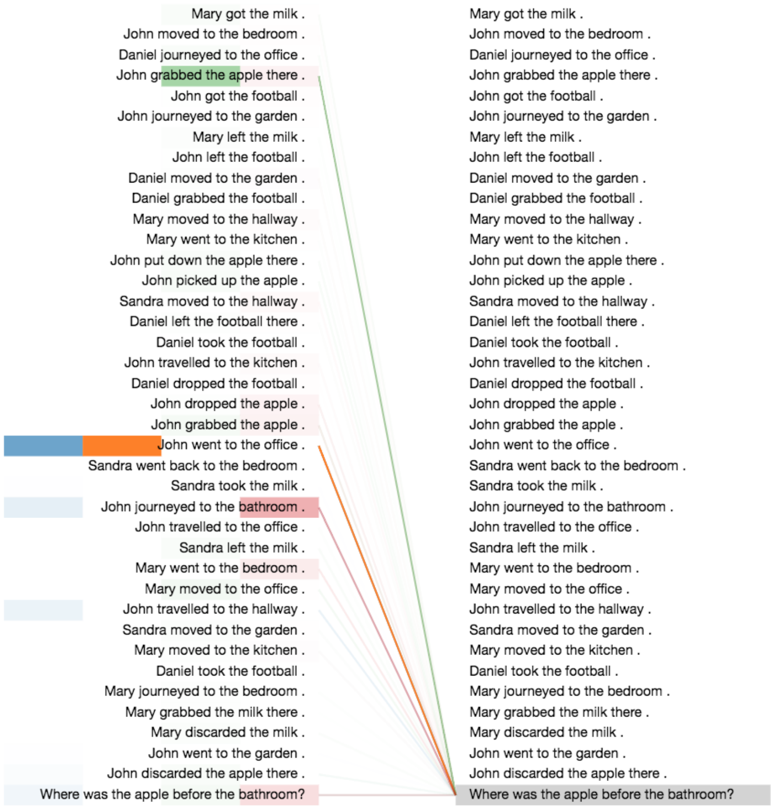
(a) Step 1


Mary got the milk .
John moved to the bedroom .
Daniel journeyed to the office .
John grabbed the apple there .
John got the football .
John journeyed to the garden .
Mary left the milk .
John left the football .
Daniel moved to the garden .
Daniel grabbed the football .
Mary moved to the hallway .
Mary went to the kitchen .
John put down the apple there .
John picked up the apple .
Sandra moved to the hallway .
Daniel left the football there .
Daniel took the football .
John travelled to the kitchen .
Daniel dropped the football .
John dropped the apple .
John grabbed the apple .
John went to the office .
Sandra went back to the bedroom .
Sandra took the milk .
John journeyed to the bathroom .
John travelled to the office .
Sandra left the milk .
Mary went to the bedroom .
Mary moved to the office .
John travelled to the hallway .
Sandra moved to the garden .
Mary moved to the kitchen .
Daniel took the football .
Mary journeyed to the bedroom .
Mary grabbed the milk there .
Mary discarded the milk .
John went to the garden .
John discarded the apple there .
Where was the apple before the bathroom?

(b) Step 2


Mary got the milk .
John moved to the bedroom .
Daniel journeyed to the office .
John grabbed the apple there .
John got the football .
John journeyed to the garden .
Mary left the milk .
John left the football .
Daniel moved to the garden .
Daniel grabbed the football .
Mary moved to the hallway .
Mary went to the kitchen .
John put down the apple there .
John picked up the apple .
Sandra moved to the hallway .
Daniel left the football there .
Daniel took the football .
John travelled to the kitchen .
Daniel dropped the football .
John dropped the apple .
John grabbed the apple .
John went to the office .
Sandra went back to the bedroom .
Sandra took the milk .
John journeyed to the bathroom .
John travelled to the office .
Sandra left the milk .
Mary went to the bedroom .
Mary moved to the office .
John travelled to the hallway .
Sandra moved to the garden .
Mary moved to the kitchen .
Daniel took the football .
Mary journeyed to the bedroom .
Mary grabbed the milk there .
Mary discarded the milk .
John went to the garden .
John discarded the apple there .
Where was the apple before the bathroom?

(c) Step 3


Mary got the milk .
John moved to the bedroom .
Daniel journeyed to the office .
John grabbed the apple there .
John got the football .
John journeyed to the garden .
Mary left the milk .
John left the football .
Daniel moved to the garden .
Daniel grabbed the football .
Mary moved to the hallway .
Mary went to the kitchen .
John put down the apple there .
John picked up the apple .
Sandra moved to the hallway .
Daniel left the football there .
Daniel took the football .
John travelled to the kitchen .
Daniel dropped the football .
John dropped the apple .
John grabbed the apple .
John went to the office .
Sandra went back to the bedroom .
Sandra took the milk .
John journeyed to the bathroom .
John travelled to the office .
Sandra left the milk .
Mary went to the bedroom .
Mary moved to the office .
John travelled to the hallway .
Sandra moved to the garden .
Mary moved to the kitchen .
Daniel took the football .
Mary journeyed to the bedroom .
Mary grabbed the milk there .
Mary discarded the milk .
John went to the garden .
John discarded the apple there .
Where was the apple before the bathroom?

(d) Step 4

Figure 6.8: Visualization of the attention distributions, when encoding the question: *"Where was the apple before the bathroom?"*.

Table 6.3: Accuracy (higher better) on the algorithmic tasks. *Note that the Neural GPU was trained with a special curriculum to obtain the perfect result, while other models are trained without any curriculum.

Model	**Copy**		**Reverse**		**Addition**	
	char-acc	*seq-acc*	*char-acc*	*seq-acc*	*char-acc*	*seq-acc*
LSTM	0.45	0.09	0.66	0.11	0.08	0.0
Transformer	0.53	0.03	0.13	0.06	0.07	0.0
Universal Transformer	0.76	0.29	0.83	0.41	0.32	0.02
UT w/ randomized offset	0.91	0.35	0.96	0.46	0.34	0.02
Neural GPU*	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

This is mainly due to recurrent inductive bias in the Universal Transformer as well as the fact that sharing parameters across depth decreases the number of parameters which helps the model generalize better.

6.4.2 Algorithmic Tasks

Generic neural network architectures cannot generalize well in algorithmic and numerical tasks requiring arithmetic operations such as addition, multiplication etc., even when they may successfully fit any given training data in such tasks, and sometimes they cannot even achieve that [283]. This can be even harder when the distribution of samples' length is different in train and test set.

We trained UTs on three algorithmic tasks, namely Copy, Reverse, and (integer) Addition, all on strings composed of decimal symbols ('0'-'9'). In all the experiments, we train the models on sequences with maximum length of 40 and evaluated on sequences with maximum length of 400 [154] to assess the ability of the models on *length generalization*. In fact, the limitation in training data in this task is the lack of coverage over all possible samples (all possible length), not the number of training samples.

As an additional inductive bias for UTs on these tasks, when calculating the positional embedding, we use positions starting with randomized offsets per sample. This way, we further encourage the model to learn position-relative transformations, which improves length generalization. Results are shown in Table 6.3. Both UT and UT with randomized position offset outperform LSTM and vanilla Transformer by a wide margin on all three tasks. The Neural GPU reports perfect results on this task [154], however, we note that this result required a special curriculum-based training protocol, e.g., training on harder (longer) samples only after the model crossed a curriculum progress threshold on easier samples (shorter).

Table 6.4: Character-level (*char-acc*) and sequence-level accuracy (*seq-acc*) results on the Memorization LTE tasks, with maximum length of 55.

	Copy		**Double**		**Reverse**	
Model	*char-acc*	*seq-acc*	*char-acc*	*seq-acc*	*char-acc*	*seq-acc*
LSTM	0.78	0.11	0.51	0.05	0.91	0.32
Transformer	0.98	0.63	0.94	0.55	0.81	0.26
Universal Transformer	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

Table 6.5: Character-level (*char-acc*) and sequence-level accuracy (*seq-acc*) results on the Program Evaluation LTE tasks with maximum nesting of 2 and length of 5.

	Program		**Control**		**Addition**	
Model	*char-acc*	*seq-acc*	*char-acc*	*seq-acc*	*char-acc*	*seq-acc*
LSTM	0.53	0.12	0.68	0.21	0.83	0.11
Transformer	0.71	0.29	0.93	0.66	**1.00**	**1.00**
Universal Transformer	**0.89**	**0.63**	**1.00**	**1.00**	**1.00**	**1.00**

6.4.3 Learning to Execute (LTE)

As another class of sequence-to-sequence learning problems, we also evaluate UTs on Learning to Execute (LTE) tasks. LTE is a set of tasks indicating the ability of a model to learn to execute computer programs and was proposed by Zaremba and Sutskever [326]. These tasks include two subsets: 1) program evaluation tasks (program, control, and addition) that are designed to assess the ability of models for understanding numerical operations, if-statements, variable assignments, the compositionality of operations, and more, as well as 2) memorization tasks (copy, double, and reverse).

The difficulty of the program evaluation tasks is parameterized by their *length* and *nesting*. The length parameter is the number of digits in the integers that appear in the programs (so the integers are chosen uniformly from [1, *length*]), and the nesting parameter is the number of times we are allowed to combine the operations with each other. Higher values of nesting yield programs with deeper parse trees. For instance, here is a program that is generated with length = 4 and nesting = 3.

```
Input:
         j=8584
         for x in range(8):
           j+=920
         b=(1500+j)
         print((b+7567))
Target:
         25011
```

We use the mix-strategy discussed in [326] to generate the datasets. Unlike [326], we do not use any curriculum learning strategy during training and we make no

Table 6.6: Accuracy on the subject-verb agreement number prediction task (higher is better).

Model	**Number of attractors**						
	0	*1*	*2*	*3*	*4*	*5*	*Total*
Previous best results [316]:							
Best Stack-RNN	*0.994*	0.979	0.965	0.935	0.916	0.880	0.992
Best LSTM	0.993	0.972	0.950	0.922	0.900	0.842	0.991
Best Attention	**0.994**	**0.977**	0.959	0.929	0.907	0.842	**0.992**
Our results:							
Transformer	0.973	0.941	0.932	0.917	0.901	0.883	0.962
Universal Transformer	0.993	0.971	**0.969**	0.940	0.921	0.892	**0.992**
UT w/ ACT	**0.994**	0.969	0.967	**0.944**	**0.932**	**0.907**	**0.992**
Δ (UT w/ ACT - Best)	0	-0.008	0.002	0.009	0.016	0.027	-

use of target sequences at test time. Tables 6.4 and 6.5 present the performance of an LSTM model, Transformer, and Universal Transformer on the program evaluation and memorization tasks, respectively. UT achieves perfect scores in all the memorization tasks and also outperforms both LSTMs and Transformers in all program evaluation tasks by a wide margin.

6.4.4 Subject-Verb Agreement

Next, we consider the task of predicting number-agreement between subjects and verbs in English sentences [178]. Succeeding in this task is a strong indicator that a model can learn to approximate syntactic structure and therefore it was proposed by Linzen et al. [178] as a proxy for assessing the ability of different models to capture hierarchical structure in natural language.

Two experimental setups were proposed by Linzen et al. [178] for training a model on this task: 1) training with a language modeling objective, i.e., next word prediction, and 2) as binary classification, i.e., predicting the number of the verb given the sentence. We follow the experimental protocol of Linzen et al. [178] for solving the task using a language modeling training setup, i.e., a next word prediction objective, followed by calculating the ranking accuracy of the target verb at test time.

In this task, in order to have different levels of difficulty, "agreement attractors" are used, i.e., one or more intervening nouns with the opposite number from the subject with the goal of confusing the model. In this case, the model needs to correctly identify the head of the syntactic subject that corresponds to a given verb and ignore the intervening attractors in order to predict the correct form of that verb. Here are some examples for this task in which subjects and the corresponding verbs are in boldface and agreement attractors are underlined: Our results are summarized in Table 6.6. The best LSTM with

```
No attractor:        The boy smiles.
One attractor:       The number of men is not clear.
Two attractors:      The ratio of men to women is not clear.
Three attractors:    The ratio of men to women and children is not clear.
```

attention from the literature achieves 99.18% on this task [316], outperforming a vanilla Transformer [282]. UTs significantly outperform standard Transformers, and achieve an *average* result comparable to the current state of the art (99.2%). However, we see that UTs (and particularly with dynamic halting) perform progressively better than all other models as the number of attractors increases (see the last row, Δ). The recurrent inductive bias, i.e., the fact that we can repeat the computations in depth, helps the Universal Transformer to capture the hierarchical relations and better model the structure of the data.

6.4.5 LAMBADA Language Modeling

The LAMBADA task [211] is a language modeling task consisting of predicting a missing target word given a broader context of 4–5 preceding sentences. The dataset was specifically designed so that humans are able to accurately predict the target word when shown the full context, but not when only shown the target sentence in which it appears. It, therefore, goes beyond language modeling, and tests the ability of a model to incorporate broader discourse and longer term context when predicting the target word.[8] Here is a sample from the dataset:

```
Context:
                 "Yes, I thought I was going to lose the baby."
                 "I was scared too," he stated, sincerity flooding his eyes.
                 "You were?"  "Yes, of course.  Why do you even ask?"
                 "This baby wasn't exactly planned for."
Target sentence:
                 "Do you honestly think that I would want you to have a ________?"
Target word:
                 miscarriage
```

The LAMBADA task consists in predicting the target word given the whole passage (i.e., the context plus the target sentence). A "control set" is also provided which was constructed by randomly sampling passages of the same shape and size as the ones used to build LAMBADA, but without filtering them in any way. The control set is used to evaluate the models at standard language modeling before testing on the LAMBADA task, and therefore to ensure that

[8] *http://clic.cimec.unitn.it/lambada/appendix_onefile.pdf*

Table 6.7: LAMBADA language modeling (LM) perplexity (lower better) with accuracy in parentheses (higher better), and Reading Comprehension (RC) accuracy results (higher better). '-' indicates no reported results in that setting. We have done the ablation study experiments (last two rows of the table) only in the language modeling setup.

Model	**LM Perplexity & (Accuracy)**			**RC Accuracy**		
	control	*dev*	*test*	*control*	*dev*	*test*
Neural Cache [112]	**129**	139	-	-	-	-
Dhingra et al. [88]	-	-	-	-	-	0.5569
Transformer	142 (0.19)	5122 (0.0)	7321 (0.0)	0.4102	0.4401	0.3988
LSTM	138 (0.23)	4966 (0.0)	5174 (0.0)	0.1103	0.2316	0.2007
UT *base*, 6 steps (fixed)	131 (0.32)	279 (0.18)	319 (0.17)	**0.4801**	0.5422	0.5216
UT w/ dynamic halting	130 (0.32)	**134** (0.22)	**142** (0.19)	0.4603	**0.5831**	**0.5625**
UT *base*, 8 steps (fixed)	129(0.32)	192 (0.21)	202 (0.18)	-	-	-
UT *base*, 9 steps (fixed)	**129(0.33)**	214 (0.21)	239 (0.17)	-	-	-

low performance on the latter cannot be attributed simply to poor language modeling.

The task is evaluated in two settings: as *language modeling* (the standard setup) and as *reading comprehension*. In the former (more challenging) case, a model is simply trained for the next-word prediction on the training data, and evaluated on the target words at test time (i.e., the model is trained to predict all words, not specifically challenging target words). In the latter setting, introduced by Chu et al. [49], the target sentence (minus the last word) is used as the query for selecting the target word from the context sentences.[9]

The results are shown in Table 6.7. Universal Transformer achieves state-of-the-art results in both the language modeling and reading comprehension setup, outperforming both LSTMs and vanilla Transformers. Note that achieving good results on the control set only shows a model's strength in standard language modeling.

Our best fixed UT results used 6 steps. However, the average number of steps that the best UT with dynamic halting took on the test data over all positions and samples was 8.2±2.1. In order to see if the dynamic model did better simply because it took more steps, we trained two fixed UT models with 8 and 9 steps respectively (see last two rows). Interestingly, these two models achieve better results compared to the model with 6 steps, but *do not outperform the UT with dynamic halting*. This leads us to believe that dynamic halting may act as a useful regularizer for the model via incentivizing smaller numbers of steps for some of the input symbols, while allowing more computation for others.

[9]Note that the target word appears in the context 81% of the time that lets selecting the word from context, which is simpler than generating it. However, the task is impossible to be solve in the remaining 19% of the cases.

Table 6.8: Machine translation results on the WMT14 En-De translation task trained on 8xP100 GPUs in comparable training setups. All *base* results have the same number of parameters.

Model	**BLEU**
Universal Transformer *small*	26.8
Transformer *base* [291]	28.0
Weighted Transformer *base* [4]	28.4
Universal Transformer *base*	**28.9**

6.4.6 Machine Translation

We trained a UT on the WMT 2014 English-German translation task in order to evaluate its performance on a large-scale sequence-to-sequence task. Results are summarized in Table 6.8.

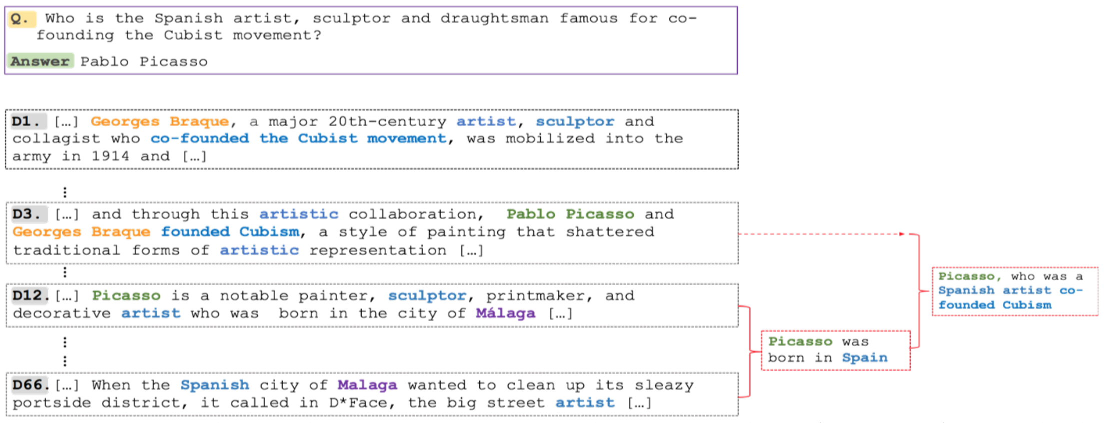
We used the same setup as reported in [291]. We trained on the standard WMT 2014 English-German dataset that consist of about 4.5 million sentence pairs. Sentences were encoded using byte-pair encoding [30] that has a shared source-target vocabulary of about 37000 tokens. During training, we used the Adam optimizer[162] with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\varepsilon = 10^{-9}$. We varied the learning rate over the course of training using similar to [291].

The UT with a fully-connected recurrent transition function (instead of separable convolution) and without ACT improves by 0.9 BLEU over a Transformer and 0.5 BLEU over a Weighted Transformer with approximately the same number of parameters [4].

6.4.7 Open-Domain Question Answering

As another real-world language understanding task, we adapt a model based on UT for the Open-domain question answering task. Open-domain question answering aims to satisfy users who are looking for a direct answer to a complex information need. This requires querying large open-domain knowledge sources like the Web. Inferring the answer to a question given multiple documents that potentially contain the answer, is at the heart of the open-domain question answering task. Most open-domain question answering systems described in the literature first retrieve relevant documents or passages, select one or a few of them as the context, and then feed the question and the context to a reading comprehension system to extract the answer [34, 41, 89, 248]. However, the information needed to answer complex questions is not always contained in a single, directly relevant document that is ranked high. In many cases, there is a need to read multiple documents, combine them, and reason over the facts from these documents to be able to give the correct answer to the question.

For example, in Figure 6.9, in order to infer the correct answer to the question:


Q. Who is the Spanish artist, sculptor and draughtsman famous for co-founding the Cubist movement?

Answer Pablo Picasso

D1. [...] Georges Braque, a major 20th-century artist, sculptor and collagist who co-founded the Cubist movement, was mobilized into the army in 1914 and [...]

⋮

D3. [...] and through this artistic collaboration, Pablo Picasso and Georges Braque founded Cubism, a style of painting that shattered traditional forms of artistic representation [...]

⋮

D12. [...] Picasso is a notable painter, sculptor, printmaker, and decorative artist who was born in the city of Málaga [...]

⋮

⋮

D66. [...] When the Spanish city of Malaga wanted to clean up its sleazy portside district, it called in D*Face, the big street artist [...]

Picasso was born in Spain

Picasso, who was a Spanish artist co-founded Cubism

Figure 6.9: Example complex question answering that requires that information from multiple documents be combined and some amount of reasoning over the information extracted from those documents (best viewed in color).

"Who is the Spanish artist, sculptor and draughtsman famous for co-founding the Cubist movement?" given the top-ranked document, a reading comprehension system most likely will extract "Georges Braque" as the answer, which is not the correct answer. In this example, in order to infer the correct answer, one has to go down the ranked list, gather and encode facts, even those that are not immediately relevant to the question, like "*Malaga is a city in Spain*," which can be inferred from a document at rank 66, and then in a multi-step reasoning process, infer some new facts, including "*Picasso was a Spanish artist*" given documents at ranks 12 and 66, and "*Picasso, who was a Spanish artist, co-founded the Cubist*" given the previously inferred fact and the document ranked third.

In this example, and in general in many cases in open-domain question answering, a piece of information in a low-ranked document that is not immediately relevant to the question, may be useful to fill in the blanks and complete information extracted from the top relevant documents and eventually support inferring the correct answer. However, most open-domain question answering methods focus on only one or a few candidate documents by filtering out the less relevant documents to avoid dealing with noisy information and operate over the selected set of documents to extract the answer [176, 297, 298].

We propose a new architecture, called TraCRNet (pronounced *Tracker Net*, that combines Transformer and Universal Transformer to improve open-domain question answering by explicitly operating on a larger set of candidate documents during the whole question answering process and learning how to aggregate and reason over information from these documents in an effective way while trying not to be distracted by noisy documents. Given the candidate documents and the question, to generate the answer, TraCRNet first **Tra**nsforms them into vectors by applying a stack of Transformer blocks with self-attention over words in each document in a layer called *Input Encoding*. Then, it updates the learned representations from the first stage by **C**ombining and enriching them through a multihop **R**easoning process by applying multiple steps of the Universal Transformer in a layer called *Multihop Reasoning*.

Returning to the example in Figure 6.9, after learning representations for each top-ranked document and the question, TraCRNet updates them by applying multiple steps of the Universal Transformer. Given the self-attention mechanism and inductive bias of the Universal Transformer, in the first step, TraCRNet can update the representation of document D#12 by attending to D#66 (as they are related by both mentioning Malaga) and augment the information in D#12 with the fact that "Malaga is city in Spain," so the updated vector of D#12 has the fact that "Picasso is a Spanish artist" encoded in itself. Then, in the next step of reasoning, TraCRNet can update the representation of D#3 by attending over the vector representing D#12 estimated in the previous step, and enrich the information in D#3 with the fact that "Picasso is a Spanish artist," and the updated vector of D#3 has the fact that "Picasso, who was a Spanish artist co-founded Cubism" encoded in it. After that, during answer generation, the decoder can attend to the final vector representing D#3 and give the correct answer.

TraCRNet has a number of desirable features. First, all the building blocks of TraCRNet are based on self-attentive feed-forward neural networks, hence per-symbol hidden state transformations are fully parallelizable, which leads to an enormous speedup during training and a super fast input encoding during inference time compared to RNN based models. Second, while there is no recurrence in time in our model, the recurrence in depth in the Universal Transformer used in the *Multihop Reasoning* layer, adds the inductive bias to the model that is needed to go beyond understanding each document separately and combine their information in multiple steps. Third, TraCRNet has the global receptive field of the Transformer based models [75, 292], which helps it to better encode a long document during *Input Encoding* as well as perform better inference over a rather large set of documents during *Multihop Reasoning*. And fourth, the hierarchical usage of a self-attention mechanism, first over words and then over documents, helps TraCRNet to control its attention both at word and document levels, making it less fragile to noisy input, which is of key importance while encoding many documents. All these properties of TraCRNet come together and lead to an effective and efficient architecture for open-domain question answering.

We employ TraCRNet on two public open-domain question answering datasets, SearchQA and Quasar-T, and achieve results that meet or exceed the state-of-the-art.

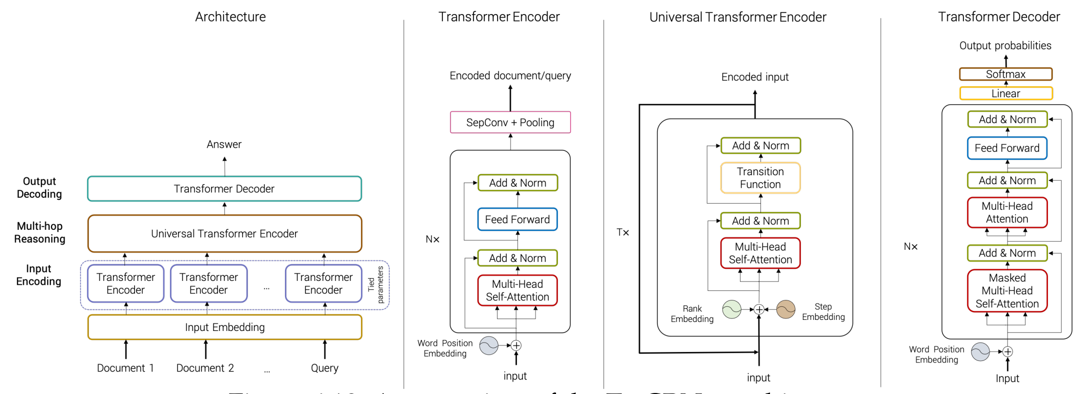


Figure 6.10: An overview of the TraCRNet architecture.

TraCRNet

In the setup we consider here, the model is given a question q and a set of n relevant documents $C_q = \{D_1^q, D_2^q, \ldots, D_n^q\}$ retrieved from the web using a search engine as the input, and the goal is to "generate" the answer a_q to the question q based on the supporting document(s) in the set C_q.

This is different from the standard Reading Comprehension (RC) tasks [134, 309]. First of all, in RC a single document (passage) is given, from which the answer should be extracted. Secondly, in RC, a strong supervision on the positions of the answer spans is available during training. We also assume that the utilized information or techniques to retrieve relevant documents are not available to the model, therefore there is no leverage for getting better-supporting documents.

TraCRNet is based on the encoder-decoder architecture, where we have a hierarchy of transformer-based models in the encoder, where the model can attend first over words and then over documents [81]. At the bottom, in the *Input Encoding* layer, we encode each document in C_q as well as the question with transformer blocks with tied parameters that are fed by word-level embeddings. Then, we feed the encoded documents and the question from this layer to the *Multihop Reasoning* layer which is, in fact, a universal transformer block where representations of all documents and the question get iteratively updated using multiple steps of self-attention. Then, we use a stack of transformer decoder blocks as the *Output Decoder* layer to generate the answer. The general schema of TraCRNet is depicted in Figure 6.10. Below, we explain the details of each of these layers in the model.

Input encoding The Input Encoding layer is in charge of encoding each of the documents and the question to single vectors given their words' embeddings. For this layer, we used a stack of N Transformer Encoder blocks that is followed by a depth-wise separable convolution [45, 153] and then a pooling function to get a single vector representation for the whole document or the question (see the `Transformer Encoder` in Figure 6.10). Depth-wise separable convolution is defined by a convolution on each of the feature channels separately, followed by a point-wise convolution that is applied to project them to a feature vector with the desirable depth (see [45] for more details).

Multihop reasoning Multihop Reasoning is the layer in which the Universal Transformer is employed to combine evidence from all documents with respect to the question within a multi-step process with the capacity of multihop reasoning. In TraCRNet, the input of the Universal Transformer Encoder is the set of vectors each representing a document in C_q or the question, that are computed by the *Input Encoding* layer (see the `Universal Transformer Encoder` in Figure 6.10).

In each step of the Universal Transformer, given $H_t \in \mathbb{R}^{(|C_q|+1)\times d}$ and the dimension d of the input vectors, we add two embeddings to H^t: a *Rank Embedding* that encodes the rank of documents given by the retrieval system, also used to distinguish the question from documents (similar to the positional embedding in token level inputs) and the *Step Embedding*. We use Equation6.8 to calculate these embeddings. In our experiments, we use depthwise separable convolution [45] as the Transition$(\cdot)$ function.

In the multihop reasoning layer, the representations of all the documents and question learned from the previous layer get updated during T steps of iterating over the Universal Transformer Encode block. Self-attention in this layer allows the model to understand each of the documents based on the information in all the documents as well as the question. In addition, the depth-wise recurrency in the Universal Transformer establishes connections among documents at each step and lays the ground for performing multihop reasoning to solve cases similar to what we have shown in Example 6.9.

Output decoder After T steps of refining the representations of documents and the question in the Universal Transformer Encoder, the final output is a matrix of d-dimensional vector representations $H \in \mathbb{R}^{(|C_q|+1)\times d}$ for all the documents in C_q and the question q.

We use a stack of N Transformer Decoder blocks (see the `Transformer Decoder` in Figure 6.10) to decode the answer. To generate answers from the model at inference time, we run the model autoregresively [113], where the model consumes the previously generated symbols at each time step in order to generate the distribution over the vocabulary for the next symbol. From this distribution, we select the symbol with the highest probability as the next symbol.

Datasets

We have conducted experiments on two publicly available open-domain question answering datasets: SearchQA [96] and Quasar-T [90]. In both of these datasets, candidate documents (passages) for each question have already been retrieved using a search engine and we do not add any extra documents to these result sets. On both datasets, human performance is evaluated in a setup where the human subjects try to find the answers to the given question from the same documents retrieved by the IR model.

SearchQA SearchQA[10] is a dataset of 140k question-answer pairs crawled from J! Archive, and augmented with text snippets retrieved using the Google search engine. For each question-answer pair, on average, about 50 web page snippets have been collected. In our experiments, we do not use the additional meta-data in the dataset like the snippet's URL.

Quasar-T Quasar-T[11] consists of 43k open-domain trivia questions and their answers obtained from various internet sources. The set of candidate documents for each question is retrieved using "Lucene" from the ClueWeb09 corpus as the background corpus. In this dataset, for each question-answer pair, a set of 100 unique passages were collected as candidate documents.

Model configuration and experimental setup

We use WordPiece embeddings [306] with a 32k token vocabulary. In both *Input Encoder* and *Output Decoder* layers, we use a stack of 6 Transformer blocks with hidden_size = 512, num_attention_heads = 8, and batch_size = 2,048. The rest of the hyper-parameters are set to the default values of the Transformer model. In the *Multihop Reasoning* layer, we have a Universal Transformer Encoder with hidden_size = 512 and num_attention_heads = 4. We set the number of recurrent steps in depth to 12. The rest of the hyper-parameters are set to the default values of the Universal Transformer model. We train with the batch size of 4,096 tokens. We use Adam with learning rate of 1×10^{-9}, $\beta_1 = 0.9$, $\beta_2 = 0.98$, L_2 weight decay of 1×10^{-04}, learning rate warmup over the first 16,000 steps, and linear decay of the learning rate. We use a dropout probability of 0.1 on all layers. Since in our model answers are generated using the decoder instead of extracting from the context, to improve the quality of generation, we pretrain all the parameters of the Transformer decoder downstream of the task of language modeling. The embeddings are shared between encoder and decoder, thus the *Input Embedding* layer also enjoys the pretraining. This helps to improve the performance especially in terms of metrics that consider the exact match of the generated answer with the ground truth. During the training of the model, we use teacher-forcing, i.e., the decoder input is the gold target, shifted to the right by one position which is the usual setup for training autoregressive models [304].

[10]*https://github.com/nyu-dl/SearchQA*
[11]*https://github.com/bdhingra/quasar*

In our experiments, TraCRNet and its variants are trained on 8 P100 GPUs for 800*k* training steps. For both datasets, a prepared version by Wang et al. [297] is used in our experiments to train and evaluate the TraCRNet as well as all the baselines. As the C_q, we consider top-50 top documents for the SearchQA, and top-100 for the Quasar-T. Following previous work on reading comprehension and open-domain question answering [34, 176, 254, 297, 298] as our evaluation metrics we adopt the F1 score, that loosely measures the average overlap between the predicted answer and the ground truth answer, and Exact Match (EM) that measures the percentage of predictions that match one of the ground truth answers exactly.[12]

Results and Discussion

Baselines We compare our results with the best reading comprehension and open-domain question answering models as well as research that achieves state-of-the-art on the SearchQA and Quasar-T datasets. To have a true apples-to-apples comparison, we only consider baselines that use no additional resources to solve the task for these datasets. We use the following methods as baselines:

1. BiDAF [248], which is a reading comprehension model with bi-directional attention flow network that uses the concatenation of top-ranked candidate documents as the context.
2. R^3 [297], which is a reinforcement learning approach that uses a ranker for selecting the most confident paragraph to train the reading comprehension model.
3. Wang et al. [298]'s model, which learns to re-rank the answers extracted by applying the R^3 model on multiple documents based on coverage and strength of each of the documents given the question.
4. Lin et al. [176]'s model, which is the most recent paper achieving state-of-the-art performance on the datasets we use for evaluation. They propose to decompose the process into a document selection to filter out noisy paragraphs, and a paragraph reader to extract the correct answer from the filtered documents. Finally, they aggregate multiple answers to obtain the final answer.

Table 6.9 presents the results of the baseline models, TraCRNet, and the human performance on both datasets.

[12]We use the tool from SQuAD [225] for evaluation.

Table 6.9: Performance of TraCRNet compared to the baseline models.

model	SearchQA		Quasar-T	
	EM	**F1**	**EM**	**F1**
BiDAF [248]	28.6	34.6	25.9	28.5
R^3 [297]	49.0	55.3	35.3	41.7
Wang et al. [298]	57.0	63.2	42.3	49.6
Lin et al. [176]	**58.8**	64.5	42.2	49.3
TraCRNet	52.9	**65.1**	**43.2**	**54.0**
Human Performance	43.9	–	51.5	60.6

Main results TraCRNet outperforms all the baselines and achieves a new state-of-the-art (to the best of our knowledge) on the Quasar-T dataset and performs as good as the best performing baseline on the SearchQA dataset. The main advantage of TraCRNet over the baselines is that it makes "full" use of the information of "all" the candidate documents in C_q. The models proposed by Lin et al. [176] and Wang et al. [298] are the strongest baselines on these datasets. Although they try to capture evidence from multiple sources by reranking or aggregating answers extracted from different documents, they filter out documents that are less likely to help at the beginning of the process. In this fashion, they lose the chance of using information from documents that are not directly relevant, like documents #12 or/and #66 in Example 6.9. However, TraCRNet keeps operating on the full set of candidate documents during the whole process and learns to what extent each document contributes to infer the final answer.

In SearchQA, we notice that for most of the questions, the answer can be extracted given a single document and in many cases, no multi-document multihop reasoning is required. Therefore, since TraCRNet *generates* the answer, as opposed to the baseline models that *extract* the answer from context, it gets a lower EM score. However, in terms of F1 score, TraCRNet slightly improves over the best baseline.

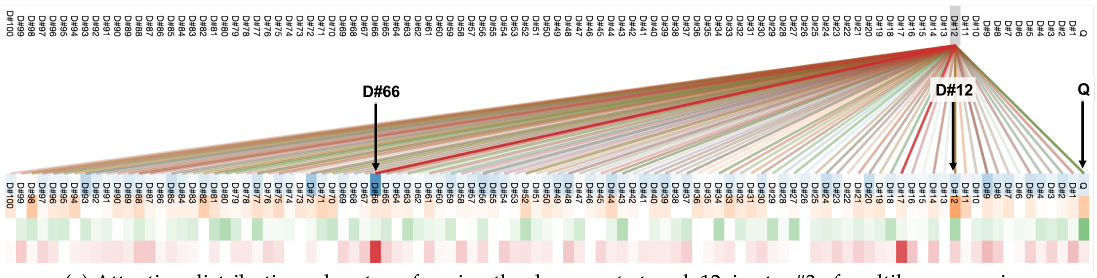
Effect of multihop reasoning In order to investigate the effect of the *Multihop Reasoning* layer, we handicap TraCRNet by removing this layer and evaluate it in two cases:

1. $\text{TraCRNet}^{\text{d}}_{\text{no-mhr}}$, in which the decoder has access to document- level representations from the encoder, and
2. $\text{TraCRNet}^{\text{w}}_{\text{no-mhr}}$ where pooling operation is removed and the decoder has access to word-level representations from the encoder.

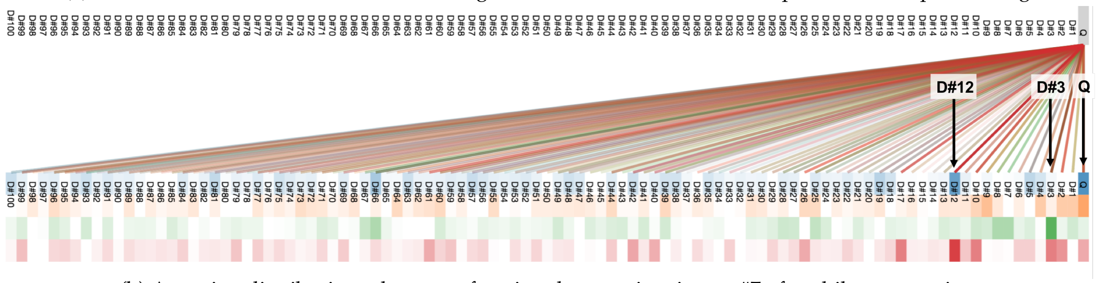
Table 6.10 presents the results of the model in these situations.

Table 6.10: Performance of TraCRNet with and without the *Multihop Reasoning* layer; numbers in parenthesis indicate percentage of performance loss.

model	**SearchQA**		**Quasar-T**	
	EM	**F1**	**EM**	**F1**
TraCRNet	52.9	65.1	43.2	54.0
TraCRNet$^{\text{d}}_{\text{no-mhr}}$	48.6 (−8%)	61.7 (−5%)	36.4 (−16%)	43.6 (−19%)
TraCRNet$^{\text{w}}_{\text{no-mhr}}$	50.2 (−5%)	59.3 (−9%)	38.1 (−12%)	40.2 (−25%)



(a) Attention distribution when transforming the document at rank 12, in step#3 of multihop reasoning.



(b) Attention distribution when transforming the question, in step#7 of multihop reasoning.

Figure 6.11: Visualization of multi-head self-attention on Multihop Reasoning layer of TraCRNet. (Best viewed in color.)

On all measures and datasets, the performance drops when we remove the *Multihop Reasoning* layer. The drop in the performance is larger on the Quasar-T dataset than on the SearchQA dataset. We noticed that trivia questions in Quasar-T, in many cases, contain clauses that should be considered together with and/or operations to be able to give the correct answer. For instance, to answer the question "What Australian food was discovered by John McAdam," we should consider that "the food is Australian" *and* "the food is discovered by John McAdam." In this situation, the chance of having multiple documents each containing one of these facts increases. Thus, having multiple supporting documents and the need for reasoning (similar to Example 6.9) will be the exact point where the advantage of the *Multihop Reasoning* layer kicks in.

Another observation here is that when we remove the *Multihop Reasoning* layer, passing word-level embeddings from the encoder to the decoder leads to better EM scores, but not to improved F1 scores. The main reason is that, in this situation, access to the input words from the decoder is more explicit. This helps the model to get closer to answer extraction than pure answer generation.

For the test example that is presented in Figure 6.9, we observed that all baseline models output "Georges Braque" which is extracted from the document at rank 1. However, unlike all the baselines, TraCRNet returns the correct answer. We looked into the attention distributions in the *Multihop Reasoning* layer of TraCRNet at different steps (of the employed Universal Transformer with 12 depth-wise recurrent steps). We were able to find a relation between attention distributions and the reasoning steps that are needed to give the correct answer to this question. We illustrate this in Figure 6.11.

Figure 6.11a presents the attention distribution over all documents and the question while encoding the document at rank 12 at step 3. TraCRNet has a high level of attention for the document at rank 66 using heads 1 and 4 (blue and red) as well as for the question using head 3 (green) while transforming the document at rank 12. This is in accordance with the fact that the model first needs to update the information encoded in the document at rank 12 with the fact that "Malaga is a city in Spain" from the document at rank 66. Later, at step 7, while encoding the question (Figure 6.11b), TraCRNet attends over document 12, which has information about "Picasso who is a Spanish artist" (updated in step 3) using heads 1 and 4 and document 3, which contains information about "Picasso as a co-founder of Cubism" using head 2 (green).

Impact of the number of documents

As we explained before, unlike most of the previous work that filters candidate documents and narrows down the set of documents under consideration to either a single document or a small set of highly relevant documents before applying an answer extractor to them, TraCRNet uses the full set of candidate documents retrieved by the search engine during the entire process of generating the answer. This is of great advantage as our analysis shows that, for some questions, the correct answer can only be extracted when considering information from low-ranked documents that are not immediately relevant to the question. However, this can potentially come at the cost of (1) efficiency, as we need to process a larger input, and of (2) performance, as there will be more noisy and non-relevant documents when we go down the ranked list of candidate documents. Making use of self-attentive feed-forward neural networks as building blocks of TraCRNet brings the ability of full per-symbol parallelization and leads to an enormous speedup on encoding the input documents. This lets the model encode a larger set of candidate documents efficiently.

To study how the performance of TraCRNet is affected by the number of candidate documents, we train and evaluate TraCRNet as well as R^3 [297] and Lin et al. [176]'s model on the Quasar-T dataset, using different numbers of

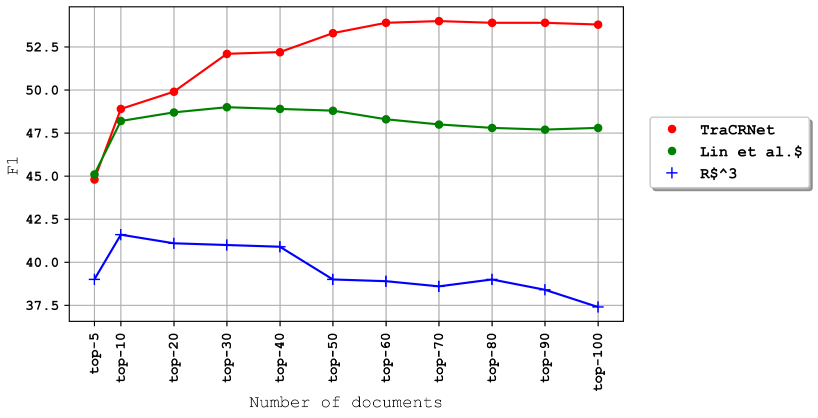


Figure 6.12: Performance in terms of F1 of TraCRNet and baselines (R^3 [297] and Lin et al. [176]'s model) with different numbers of candidate documents on Quasar-T dataset.

candidate documents associated with each question.[13] Figure 6.12 presents the performance of these models when they are fed with the top-5, top-10, . . . , top-100 retrieved documents. As can be seen, although Lin et al. [176]'s model is pretty good at staying robust when noise increases (it is designed to learn from distant supervision), increasing the number of candidate documents eventually leads to a small drop in performance of both baselines due to the noise in the low-ranked documents. However, TraCRNet not only controls the effect of noisy low-ranked documents by calibrating their effect on inferring the final answer through self-attention, but it also keeps improving as we increase the number of documents as it can exploit any useful information contained in low-ranked documents which can help better understand the question or perform reasoning.

6.5 Conclusion

In this chapter we focused on addressing **RQ-3.1**: *"How can we improve the generalization and data efficiency of self-attentive feed-forward sequence models by injecting a recurrent inductive bias?"* We introduced the Universal Transformer to address **RQ-3.1.1**, a generalization of the Transformer model that extends its theoretical capabilities, by introducing the recurrent inductive bias in depth. The Universal Transformer addresses a key shortcoming of the standard Transformer. It combines the following key properties into one model:

Weight sharing: Following intuitions behind weight sharing found in CNNs and RNNs, we extend the Transformer with a simple form of weight sharing

[13]In this experiment, we just change the initial number of candidates, but we train baseline models with their original setups and do not impose any assumption (e.g., fixing the candidate list) on them.

that strikes an effective balance between inductive bias and model expressivity, which we show extensively on both small and large-scale experiments.

Conditional computation: In our goal to build a computationally universal machine, we equipped the Universal Transformer with the ability to halt or continue computation through the ACT mechanism, which shows stronger results compared to the fixed-depth Universal Transformer.

We have employed the Universal Transformer on a wide range of challenging sequence modeling tasks to address **RQ-3.1.2**. We showed that the Universal Transformer as an strong data-efficient model achieves state-of-the-art results on the bAbI tasks, which is a set of language understanding a reasoning tasks, with limited number of data. We also evaluated the UT on a set of algorithmic tasks in which a strong length generalization is needed and showed that with inductive biases injected in the model, we can improve the generalization. We also evaluated the UT on real-world tasks like machine translation, question answering, and broad context language modeling, where it achieves strong results.

In Part III of the thesis, we focused on addressing **RQ-3**: *"How can inductive biases help to improve the generalization and data efficiency of learning algorithms?"*. we explored the idea of injecting inductive biases into learning algorithms to improve their data-efficiency and generalization. This is, in a sense, defining an "innate" abstract knowledge for the learning algorithms that can help overcome the challenging problem of the poverty of stimulus.

7 Conclusion

The success of today's supervised machine learning algorithms in complex tasks depends strongly on the availability of large scale high quality labeled data. In practice, however, for many applications and domains, the available training data is limited or noisy and it is difficult to build machines that can learn with such imperfect training data. In contrast, humans are capable of uncovering the underlying concepts, relations, and structure of sparsely observed data with variable quality and use that knowledge to go far beyond the scarcity of the data and routinely make successful generalizations based on them.

The argument of the *poverty of stimulus* [48] in human learning suggests that the observed data is not rich enough for selecting a correct target hypothesis without postulating an a priori knowledge [47]. In line with this, when designing machine learning algorithms, pure *data-driven* learning, which relies only on previous experience, does not seem to be able to learn generalizable solutions [198]. Similar to human's *innately primed* learning, having part of the knowledge encoded in the learning algorithms in the form of strong or weak biases, can help them learn solutions that better generalize to unseen samples [199].

In this thesis, we focused on this problem of the poverty of stimulus for learning algorithms. We argued that even noisy and limited signals can contain a great deal of valid information that can be incorporated along with prior knowledge and biases that are encoded into learning algorithms in order to solve complex problems. We study how to improve the learning with imperfect supervision signals in the context of language understanding and sequence modeling tasks.

7.1 Research Questions and Conclusions

We referred to *"imperfect supervision"* as a general term covering a variety of situations where the learning process is based on imperfect training examples. This imperfection can be in the number or coverage of training examples like learning from *incomplete supervision*, where only a limited subset of data is labeled or no labeled data is available. The imperfection can also refer to the labeling process like in *inexact supervision* where only coarse-grained annotations are provided or *inaccurate supervision* where the given labels are noisy and they are not always ground truth [334]. We also consider situations where not only the labels in the training data but also feature vectors can be limited, noisy, or subject to change over time.

We formulated the main research question of the thesis as:

> **RQ-Main** *How can we improve the learning process for language understanding tasks, if the supervision signal is noisy in quality or limited in quantity?*

We broke down our main research question into three questions and addressed each of them in each part of this thesis. The first question, that we addressed in Part I of the thesis, was:

> **RQ-1** *How to use the structure of the data as prior knowledge to learn robust and effective representations of entities and concepts, when the data is noisy or variable over time?*

In Part I, we focused on learning representations for entities and abstract concepts, like topical relevance, a political conviction, etc., given data in which the features and labels can be noisy or subject to change over time.

First, in Chapter 2 informed by a discussion of the early work by Luhn [183] about *significant words* we introduced *Significant Words Language Models (SWLMs)* that estimate a representation for an entity or concept that is associated with a set of textual documents in a way that the estimated representation captures significant features by avoiding the distracting effect of common features as well as rare features. We evaluated SWLMs on a set of problems including (pseudo-)relevance feedback in document ranking and group profiling in contextual suggestion and recommendation, and showed that we can improve the quality and robustness of representations by making them dependent less on general or specific features, but rely more on significant features.

Then, in Chapter 3, we extended our discussions in Chapter 2 to hierarchical structures and introduced *Hierarchical Significant Words Language Models*

(HSWLMs) for estimating separable representations for hierarchical entities. We demonstrated that based on ranking and classification principles, the *separation property* in the data representation is a desirable foundational property which leads to separability of scores and consequently improves the accuracy of classifiers' decisions.

We showed that in order to have horizontally and vertically separable representations for hierarchically structured data, they should capture all, and only, the essential features of the entities taking their position in the hierarchy into account, which is the key idea behind HSWLMs. We evaluated the performance of classification over time using separable representations learned by HSWLMs and showed that separability makes the model more robust and transferable over time by filtering out non-essential and non-stable features.

The main conclusion of Part I is that incorporating prior knowledge can help to improve the robustness of the outcome of the learning process in noisy and variable environments. We showed that taking the general structure of the data as prior knowledge, which is, in fact, a form of inductive bias, can help to learn representations that are not only effective but also less affected by noisy factors in the data.

The second question, which we addressed in Part II of the thesis, was:

> **RQ-2** *How to design learning algorithms that can learn from weakly annotated samples, while generalizing over the imperfection in their labels?*

In Part II, we focused on how to augment the training data using a vast amount of data that are not hand-labeled, but weakly annotated using, for instance, a heuristic function. We discuss how to train learning algorithms using such weak labels and how to learn properties of the weakly annotated data, like the quality of labels and incorporate those in the learning process.

First, in Chapter 4, we proposed to use unsupervised methods in order to programmatically generate large amounts of training data [226], as weakly annotated data, to train effective neural ranking models. We focus on the task of assessing the topical relevance, i.e., ranking documents given a query. We examined various neural ranking models with different ranking architectures and objectives, and different input representations.

We found that providing the network with raw data and letting the network learn the features that matter, gives the network a chance of learning how to ignore imperfection in the training data. Also we showed in the case of having weakly annotated training data, by targeting some explicit labels from the data, we may end up with a model that has learned to express the data very well, but is incapable of going beyond it. We also observed that when learning from weakly annotated data, it is crucial to provide the network with a considerable amount of diverse training examples to help the model learn at the edge of its capacity.

Then, in Chapter 5, we proposed a set of systematic approaches that are tasks and architecture independent and can meta-learn the quality of the labels and explicitly control the learning process with respect to the estimated qualities. We introduced two semi-supervised learning approaches in the presence of weakly labeled data: *Learning from Controlled Weak Supervision (CWS)* and *fidelity-weighted learning (FWL)*. CWS is a meta-learning approach that unifies learning to estimate the confidence score of weak annotations and training neural networks to learn a target task with controlled weak supervision, i.e., using weak labels to update the parameters but taking their estimated confidence scores into account. FWL is a student-teacher framework in which the student network is in charge of learning a target task given a vast amount of samples with weak labels associated with fidelity scores that are generated by the teacher network. We applied both CWS and FWL to document ranking and sentiment classification and empirically verified that they improve the learning process in terms of performance and convergence time.

The main conclusion of part II is that we can design models that are capable of learning from weakly annotated data by defining proper training objectives. We also found that we can use the data to meta-learn some properties of the data, like the quality of labels and incorporate these properties in the learning process.

The last question, that we addressed in Part III of the thesis, was:

> **RQ-3** *How can inductive biases help to improve the generalization and data efficiency of learning algorithms?*

In Part III, we focused on sequence processing neural networks and studied the role of inductive biases, like recurrent inductive bias, on the generalization and data efficiency of these models on different language understanding and sequence modeling tasks.

In Chapter 6, we argued that the lack of recurrent inductive bias in feed-forward self-attentive models, like the Transformer, can lead to the failure of the model on complex reasoning tasks with limited data, algorithmic tasks where length generalization over training samples is needed, and structured language understanding tasks. We proposed the Universal Transformer, a self-attentive concurrent-recurrent sequence model, in which we introduce recurrence in depth by repeatedly modifying a series of vector representations for each position of the sequence in parallel. The key idea of the universal transformer is sharing the parameters across the layers which forms a recurrent inductive bias in depth and saves massively on the number of parameters and leads to a more data efficient model.

We also discussed other assumptions whose encoding them in a model as inductive biases can help to extrapolate from training data and to better generalizing at test time. We also introduced variants of the Universal Transformer with conditional computations and showed that it can achieve stronger results compared to the fixed-depth Universal Transformer.

The main conclusion of Part III is that injecting inductive biases into learning algorithms can improve their data-efficiency and generalization. These inductive biases are in fact innate prior knowledge for the learning algorithms that can eventually help overcoming the challenging problem of the poverty of stimulus.

7.1.1 Contributions

Here is a list of main contributions of this thesis to solving problems in information retrieval, natural language processing, and machine learning:

- We proposed approaches for learning *robust* and *time agnostic* representations for documents, given the relations in the data as prior knowledge.

- We presented a new family of models, significant words language models, that capture only and all essential features for representing a set of documents and showed they are not only easily inspectable by human, but also effective in many tasks, like classification, (pseudo)-relevance feedback for document ranking, and contextual suggestion.

- We proposed using heuristic unsupervised models as weak labelers to create a large-scale weakly annotated training set, for tasks where no such training set is available like document ranking. We further studied the effectiveness of different general architectures for neural rankers and various objective functions when learning with weak supervision. This provided excellent groundwork for many researchers to apply deep neural networks on search and ranking problems.

- We proposed approaches for learning to learn from weak supervision, by jointly optimizing for the objectives of the main task as well as learning and incorporating the fidelity of labels. The proposed ideas have been used already by many other researchers on applications/tasks where the labels in the training set are of variable qualities.

- We proposed a Turing complete version of the transformer model, the universal transformer, in which we introduce recurrence-in-depth that increases generalization and effectiveness of the model due to the introduced recurrent inductive bias. We also used adaptive computation and sowed effectiveness of efficiency of the model in several sequence modeling tasks.

The **general conclusion** of this thesis is that there are a number of effective approaches to improve the process of learning with imperfect supervision. Specifically, by (i) *employing prior knowledge in learning algorithms* (Part I), (ii) *augmenting data and learning to learn how to better use the data* (Part II), and (iii) *introducing inductive biases to learning algorithms* (Part III).

More generally, we focused on language understanding tasks, like assessing relevance on textual documents, machine translation, question answering, and natural language reasoning. In these tasks, to address the problem of imperfect supervision, i,e, when the training data or training labels are noisy in quality or limited in quantity, we developed ideas that result in a better "product", i.e better performance in terms of prediction outcomes. However they also result in a better learning "process" resulting in better models that are able to extract valuable cues from imperfect signals and capture key aspects of the task at hand.

7.2 Towards Building Machines that Deal with the Poverty of Stimulus

There has been a long discussion between empiricists and rationalists in the context of human learning concerning the extent to which we are dependent on our experiences and observations in our effort to gain knowledge [190]. Empiricists claim that sensory observations (data) are the ultimate source of all our concepts and knowledge, while rationalists claim that there are significant ways in which our concepts and knowledge are gained independently of our experiences and observations.

A similar discussion has been raised in machine learning in the context of learning algorithms. Some machine learning researchers believe that the intrinsic complexity of the world means we should not build any prior knowledge into our systems: "seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation."[1]

[1] "The Bitter Lesson" by Rich Sutton: *http://www.incompleteideas.net/IncIdeas/BitterLesson.html*

Other machine learning researchers, on the other hand, emphasize the fact that "there are no predictions without assumptions, no generalization without inductive bias" and believe that the complexity of the world, as a matter of fact, leads to crippling intractability for the approaches on which empiricists proposes to rely and argue that only with the right prior knowledge and the right inductive biases, we can get a handle on that complexity.[2]

In practice, we do not apply either rigorous empiricist or rationalist approaches in machine learning. When we pursue more explanatory rationalist approaches, we always recognize the importance of observations and the data as the means by which reality affects our understanding. When taking a data-driven approach, we make use of reasoning at least in the act of observing data, i.e., choices such as how data is selected, assumptions we make, biases to our algorithms, and the overall architectures of our solution. In order to build machines that can create knowledge a truce between these two main approaches is essential. There is no doubt that the success of deep learning is very much a success of scale and in general, machine learning methods that survive the test of time and make breakthrough progress tend to scale. But incorporating knowledge or inductive biases has its own success stories and the question here is more about "what" that knowledge or biases should be and "when" and "how" it should be incorporated.

While finding the right inductive biases is hard, they can enable progress on intractable problems or situations where we cannot rely on arbitrarily scaling of computation such as settings with noisy or limited data, which characterizes most real-world applications. Besides, scalability can be defined in many dimensions, if a method is "scalable" with more data and computation, it has a chance of succeeding only in a subset of problems that we can gather infinite data for them. However, we can define "scale," as in "scale to new problems," which is, in fact, the ability of generalization, where inductive biases can play crucial roles to achieve it, especially when considering the problem of poverty of stimulus.

We are enthusiastic about recent developments in machine learning models that are able to learn with imperfect supervision. By combining ideas on how to incorporate general prior knowledge, how to better use the data and meta learn its properties, and how to inject the right inductive biases, we hope that further improvements presented in this thesis will help us build learning algorithms that are more powerful, more data efficient, and more generalizable, and in a bigger picture, help machines to get closer to human-level intelligence.

[2]"Do we still need models or just more data and compute?" by Max Welling: *https://staff.fnwi.uva.nl/m.welling/wp-content/uploads/Model-versus-Data-AI-1.pdf*

Bibliography

[1] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM.

[2] Abdul-jaleel, N., Allan, J., Croft, W. B., Diaz, O., Larkey, L., Li, X., Smucker, M. D., and Wade, C. (2004). Umass at trec 2004: Novelty and hard. In *TREC-13*.

[3] Abrantes, P. (1999). Analogical reasoning and modeling in the sciences. *Foundations of Science*, 4(3):237–270.

[4] Ahmed, K., Keskar, N. S., and Socher, R. (2017). Weighted transformer network for machine translation. *arXiv preprint arXiv:1711.02132*.

[5] Amer-Yahia, S., Roy, S. B., Chawlat, A., Das, G., and Yu, C. (2009). Group recommendation: Semantics and efficiency. *VLDB*, 2:754–765.

[6] Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., and de Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989.

[7] Arampatzis, A., Kamps, J., and Robertson, S. (2009). Where to stop reading a ranked list?: Threshold optimization using truncated score distributions. In *SIGIR '09*, pages 524–531.

[8] Arampatzis, A. and van Hameran, A. (2001). The score-distributional threshold optimization for adaptive binary classification tasks. In *SIGIR '01*, pages 285–293.

[9] Ardissono, L., Goy, A., Petrone, G., Segnan, M., and Torasso, P. (2003). Intrigue: personalized recommendation of tourist attractions for desktop and hand held devices. *Applied Artificial Intelligence*, 17(8-9):687–714.

[10] Asadi, N., Metzler, D., Elsayed, T., and Lin, J. (2011). Pseudo test collections for learning web search ranking functions. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 1073–1082. ACM.

[11] Azarbonyad, H., Dehghani, M., Beelen, K., Arkut, A., Marx, M., and Kamps, J. (2017a). Words are malleable: Computing semantic shifts in political and media discourse. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17.

[12] Azarbonyad, H., Dehghani, M., Kenter, T., Marx, M., Kamps, J., and de Rijke, M. (2018). HiTR: Hierarchical topic model re-estimation for measuring topical diversity of documents. *IEEE Transactions on Knowledge and Data Engineering*.

[13] Azarbonyad, H., Dehghani, M., Kenter, T., Marx, M., Kamps, J., and de Rijke, M. (2017b). Hierarchical re-estimation of topic models for measuring topical diversity. In *European Conference on Information Retrieval (ECIR'17)*.

[14] Azarbonyad, H., Dehghani, M., Marx, M., and Kamps, J. (2015a). Time-aware authorship attribution for short text streams. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15.

[15] Azarbonyad, H., Dehghani, M., Marx, M., and Kamps, J. (2020). Learning to rank for multi label text classification: Combining different sources of information. In *Journal of Natural Language Engineering*.

[16] Azarbonyad, H., Saan, F., Dehghani, M., Marx, M., and Kamps, J. (2015b). Are topically diverse documents also interesting? In *International Conference of the Cross-Language Evaluation Forum for European Languages (CLEF)*.

[17] Ba, J. and Caruana, R. (2014). Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662.

[18] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

[19] Baccianella, S., Esuli, A., and Sebastiani, F. (2010). Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204.

[20] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

[21] Ballard, D. H. (2015). *Brain Computation As Hierarchical Abstraction*. The MIT Press.

[22] Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

[23] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

[24] Bing, L., Chaudhari, S., Wang, R. C., and Cohen, W. W. (2015). Improving distant supervision for information extraction using label propagation through lists. In *EMNLP '15*, pages 524–529.

[25] Blei, D. M. and Lafferty, J. D. (2006). Dynamic topic models. In *ICML*, pages 113–120.

[26] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 92–100.

[27] Borisov, A., Markov, I., de Rijke, M., and Serdyukov, P. (2016). A neural click model for web search. In *WWW '16*, pages 531–541.

[28] Botvinick, M. M. (2008). Hierarchical models of behavior and prefrontal function. *Trends in cognitive sciences*, 12(5):201.

[29] Brank, J., Grobelnik, M., Milic-Frayling, N., and Mladenic, D. (2002). Feature selection using linear support vector machines. Technical Report MSR-TR-2002-63, Microsoft Research.

[30] Britz, D., Goldie, A., Luong, M.-T., and Le, Q. (2017). Massive exploration of neural machine translation architectures. *arXiv preprint arXiv:1703.03906*.

[31] Brodley, C. E. and Friedl, M. A. (1999). Identifying mislabeled training data. *Journal of artificial intelligence research*, 11:131–167.

[32] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1993). Signature verification using a "siamese" time delay neural network. In *NIPS '93*, pages 737–744.

[33] Bucilua, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM.

[34] Buck, C., Bulian, J., Ciaramita, M., Gajewski, W., Gesmundo, A., Houlsby, N., and Wang, W. (2018). Ask the right questions: Active question reformulation with reinforcement learning. In *International Conference on Learning Representations*.

[35] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005). Learning to rank using gradient descent. In *ICML '05*, pages 89–96.

[36] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167.

[37] Carpineto, C. and Romano, G. (2012). A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.*, 44(1):1:1–1:50.

[38] Carpineto, C. and Romano, G. (2013). Semantic search log k-anonymization with generalized k-cores of query concept graph. In *ECIR'13*, pages 110–121.

[39] Chang, M.-W., Srikumar, V., Goldwasser, D., and Roth, D. (2010). Structured output learning with indirect supervision. In *Proceedings of the 27th International Conference on Machine Learning*, ICML'10, pages 199–206.

[40] Chapelle, O., Schölkopf, B., and Zien, A. (2006). *Semi-Supervised Learning*. The MIT Press, 1st edition.

[41] Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017). Reading wikipedia to answer open-domain questions. In *55th Annual Meeting of the Association for Computational Linguistics*, pages 1870–1879.

[42] Chen, M., Weinberger, K. Q., and Blitzer, J. (2011). Co-training for domain adaptation. In *Advances in Neural Information Processing Systems 24*, pages 2456–2464.

[43] Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

[44] Chollet, F. (2016). Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*.

[45] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Conference on Computer Vision and Pattern Recognition*.

[46] Chomsky, N. (1965). *Aspects of the Theory of Syntax*. The MIT Press, Cambridge.

[47] Chomsky, N. (1971). *Problems of knowledge and freedom: The Russell lectures*. Courier Corporation.

[48] Chomsky, N. (1980). Rules and representations. *Behavioral and brain sciences*, 3(1):1–15.

[49] Chu, Z., Wang, H., Gimpel, K., and McAllester, D. (2017). Broad context language modeling as reading comprehension. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 52–57.

[50] Cirillo, C., Chang, Y., and Razon, J. (1969). Evaluation of feedback retrieval using modified freezing, residual collection, and test and control groups. *Scientific Report No. ISR-16 to the National Science Foundation*.

[51] Clarke, J., Goldwasser, D., Chang, M.-W., and Roth, D. (2010). Driving semantic parsing from the world's response. In *Proceedings of the fourteenth conference on computational natural language learning*, pages 18–27.

[52] Cohen, D. and Croft, W. B. (2016). End to end long short term memory networks for non-factoid question answering. In *ICTIR '16*, pages 143–146.

[53] Cohen, T. and Welling, M. (2016). Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999.

[54] Cohen, T. S. and Welling, M. (2017). Steerable cnns. In *International Conference on Learning Representations*.

[55] Collins-Thompson, K. and Callan, J. (2007). Estimation and use of uncertainty in pseudo-relevance feedback. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 303–310.

[56] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

[57] Cormack, G. V., Smucker, M. D., and Clarke, C. L. (2011). Efficient and effective spam filtering and re-ranking for large web datasets. *Inf. Retr.*, 14(5):441–465.

[58] Crestani, F., Lalmas, M., Van Rijsbergen, C. J., and Campbell, I. (1998). "is this document relevant?... probably...": A survey of probabilistic models in information retrieval. *ACM Comput. Surv.*, 30(4):528–552.

[59] Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2018). Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*.

[60] Custers, B. (2003). Effects of unreliable group profiling by means of data mining. In *Discovery Science*, pages 291–296.

[61] de Swaan, A. (1973). *Coalition Theories and Cabinet Formations: A Study of Formal Theories of Coalition Formation Applied to Nine European Parliaments after 1918*, volume 4 of *Progress in Mathematical Social Sciences*. Elsevier, New York.

[62] Dehghani, M. (2016). Significant words representations of entities. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16.

[63] Dehghani, M. (2018). Toward document understanding for information retrieval. *ACM SIGIR Forum*, 51(3).

[64] Dehghani, M., Abnar, S., and Kamps, J. (2016a). The healing power of poison: Helpful non-relevant documents in feedback. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16.

[65] Dehghani, M., Azarbonyad, H., Kamps, J., Hiemstra, D., and Marx, M. (2016b). Inoculating relevance feedback against poison pills. In *15th Dutch-Belgian Information Retrieval Workshop, DIR 2016*.

[66] Dehghani, M., Azarbonyad, H., Kamps, J., Hiemstra, D., and Marx, M. (2016c). Luhn revisited: Significant words language models. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16.

[67] Dehghani, M., Azarbonyad, H., Kamps, J., and Marx, M. (2016d). Generalized group profiling for content customization. In *CHIIR '16*, CHIIR '16.

[68] Dehghani, M., Azarbonyad, H., Kamps, J., and Marx, M. (2016e). On horizontal and vertical separation in hierarchical text classification. In *The proceedings of ACM SIGIR International Conference on the Theory of Information Retrieval*, ICTIR'16.

[69] Dehghani, M., Azarbonyad, H., Kamps, J., and Marx, M. (2016f). Significant words language models for contextual suggestion. *Proceedings National Institute for Standards and Technology. NIST Special Publication: SP*, 500.

[70] Dehghani, M., Azarbonyad, H., Kamps, J., and Marx, M. (2016g). Two-way parsimonious classification models for evolving hierarchies. In *Proceedings of Conference and Labs of the Evaluation Forum*, CLEF '16.

[71] Dehghani, M., Azarbonyad, H., Kamps, J., and de Rijke, M. (2017a). Share your model instead of your data: Privacy preserving mimic learning for ranking. In *SIGIR Workshop on Neural Information Retrieval*, SIGIR-NeuIR'17.

[72] Dehghani, M., Azarbonyad, H., Kamps, J., and de Rijke, M. (2019a). Learning to transform, combine, and reason in open-domain question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19.

[73] Dehghani, M., Azarbonyad, H., Marx, M., and Kamps, J. (2015a). Learning to combine sources of evidence for indexing political texts. In *Proceedings of the Dutch-Belgian Information Retrieval Workshop*.

[74] Dehghani, M., Azarbonyad, H., Marx, M., and Kamps, J. (2015b). Sources of evidence for automatic indexing of political texts. In *Proceedings of European Conference on IR Research*, ECIR'15.

[75] Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, Ł. (2019b). Universal transformers. In *International Conference on Learning Representations*, ICLR'19.

[76] Dehghani, M., Jagfeld, G., Azarbonyad, H., Olieman, A., Kamps, J., and Marx, M. (2017b). On search powered navigation. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '17.

[77] Dehghani, M., Jagfeld, G., Azarbonyad, H., Olieman, A., Kamps, J., and Marx, M. (2017c). Telling how to narrow it down: Browsing path recommendation for exploratory search. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*, CHIIR '17.

[78] Dehghani, M. and Kamps, J. (2018). Learning to rank from samples of variable quality. In *SIGIR 2018 Workshop on Learning from Limited or Noisy Data for Information Retrieval*.

[79] Dehghani, M., Mehrjou, A., Gouws, S., Kamps, J., and Schölkopf, B. (2018). Fidelity-weighted learning. In *International Conference on Learning Representations*, ICLR'18.

[80] Dehghani, M., Mehrjou, A., Gouws, S., Kamps, J., and Schölkopf, B. (2019c). Learning from samples of variable quality. In *ICLR workshop on Learning from Limited Labeled Data*, ICLR-LLD'19.

[81] Dehghani, M., Rothe, S., Alfonseca, E., and Fleury, P. (2017d). Learning to attend, copy, and generate for session-based query suggestion. In *Proceedings of The international Conference on Information and Knowledge Management*, CIKM'17.

[82] Dehghani, M., Severyn, A., Rothe, S., and Kamps, J. (2017e). Avoiding your teacher's mistakes: Training neural networks with controlled weak supervision. *arXiv preprint arXiv:1711.00313*.

[83] Dehghani, M., Severyn, A., Rothe, S., and Kamps, J. (2017f). Learning to learn from weak supervision by full supervision. In *NIPS2017 workshop on Meta-Learning*, NIPS-MetaLearn'17.

[84] Dehghani, M., Zamani, H., Severyn, A., Kamps, J., and Croft, W. B. (2017g). Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17.

[85] Deriu, J., Gonzenbach, M., Uzdilli, F., Lucchi, A., De Luca, V., and Jaggi, M. (2016). Swisscheese at semeval-2016 task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision. *Proceedings of SemEval*, pages 1124–1128.

[86] Deriu, J., Lucchi, A., De Luca, V., Severyn, A., Müller, S., Cieliebak, M., Hofmann, T., and Jaggi, M. (2017). Leveraging large amounts of weakly supervised data for multi-language sentiment classification. In *Proceedings of the 26th international International World Wide Web Conference*, WWW'17, pages 1045–1052.

[87] Desautels, T., Krause, A., and Burdick, J. W. (2014). Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *Journal of Machine Learning Research*, 15(1):3873–3923.

[88] Dhingra, B., Jin, Q., Yang, Z., Cohen, W. W., and Salakhutdinov, R. (2018). Neural models for reasoning over multiple mentions using coreference. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 42–48.

[89] Dhingra, B., Liu, H., Yang, Z., Cohen, W. W., and Salakhutdinov, R. (2017a). Gated-attention readers for text comprehension. In *55th Annual Meeting of the Association for Computational Linguistics*, pages 1832–1846.

[90] Dhingra, B., Mazaitis, K., and Cohen, W. W. (2017b). Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

[91] Dhingra, B., Yang, Z., Cohen, W. W., and Salakhutdinov, R. (2017c). Linguistic knowledge as memory for recurrent neural networks. *arXiv preprint arXiv:1703.02620*.

[92] Diaz, F. (2016). Learning to rank with labeled features. In *ICTIR '16*, pages 41–44.

[93] Diaz, F., Mitra, B., and Craswell, N. (2016). Query Expansion with Locally-Trained Word Embeddings. In *Proceedings of Association for Computational Linguistics*.

[94] Donahue, J., Krähenbühl, P., and Darrell, T. (2017). Adversarial feature learning. In *ICLR2017*.

[95] Dosovitskiy, A., Fischer, P., Springenberg, J. T., Riedmiller, M., and Brox, T. (2016). Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1734–1747.

[96] Dunn, M., Sagun, L., Higgins, M., Guney, V. U., Cirik, V., and Cho, K. (2017). Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

[97] Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625–660.

[98] et al., M. A. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[99] Finn, C., Abbeel, P., and Levine, S. (2017a). Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org.

[100] Finn, C., Abbeel, P., and Levine, S. (2017b). Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

[101] Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3:1289–1305.

[102] Fredrikson, M., Jha, S., and Ristenpart, T. (2015). Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333.

[103] Frénay, B. and Verleysen, M. (2014). Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869.

[104] Gabrilovich, E., Smola, A., and Tishby, T. (2010). Feature generation and selection for information retrieval. In *Workshop of SIGIR, 2010*.

[105] Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122.

[106] Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58.

[107] Gentner, D. and Markman, A. B. (1997). Structure mapping in analogy and similarity. *American psychologist*, 52(1):45.

[108] Go, A., Bhayani, R., and Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12).

[109] Goldberger, J. and Ben-Reuven, E. (2017). Training deep neural-networks using a noise adaptation layer. In *ICLR2017*.

[110] Goodwin, G. P. and Johnson-Laird, P. (2005). Reasoning about relations. *Psychological review*, 112(2):468.

[111] Gopal, S. and Yang, Y. (2013). Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *SIGKDD*, pages 257–265.

[112] Grave, E., Joulin, A., and Usunier, N. (2016). Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*.

[113] Graves, A. (2013). Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

[114] Graves, A. (2016). Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*.

[115] Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *CoRR*, abs/1410.5401.

[116] Griffiths, T. L., Chater, N., Kemp, C., Perfors, A., and Tenenbaum, J. B. (2010). Probabilistic models of cognition: Exploring representations and inductive biases. *Trends in cognitive sciences*, 14(8):357–364.

[117] Gulcehre, C., Denil, M., Malinowski, M., Razavi, A., Pascanu, R., Hermann, K. M., Battaglia, P., Bapst, V., Raposo, D., Santoro, A., et al. (2018). Hyperbolic attention networks. *arXiv preprint arXiv:1805.09786*.

[118] Guo, J., Fan, Y., Ai, Q., and Croft, W. B. (2016). A deep relevance matching model for ad-hoc retrieval. In *CIKM '16*, pages 55–64.

[119] Guo, J., Fan, Y., Pang, L., Yang, L., Ai, Q., Zamani, H., Wu, C., Croft, W. B., and Cheng, X. (2019). A deep look into neural ranking models for information retrieval. *arXiv preprint arXiv:1903.06902*.

[120] Ha-Thuc, V. and Renders, J.-M. (2011). Large-scale hierarchical text classification without labelled data. In *WSDM*, pages 685–694.

[121] Halevy, A., Norvig, P., and Pereira, F. (2009). The unreasonable effectiveness of data. *IEEE Intelligent Systems*.

[122] Hamdan, H., Béchet, F., and Bellot, P. (2013). Experiments with dbpedia, wordnet and sentiwordnet as resources for sentiment analysis in microblogging. In *Second Joint Conference on Lexical and Computational Semantics*, volume 2, pages 455–459.

[123] Han, X. and Sun, L. (2016). Global distant supervision for relation extraction. In *AAAI'16*, pages 2950–2956.

[124] Harman, D. (1992). *Information Retrieval*, chapter Relevance Feedback and Other Query Modification Techniques, pages 241–263. Prentice-Hall, Inc.

[125] Harman, D. and Buckley, C. (2009). Overview of the reliable information access workshop. *Inf. Retr.*, 12(6):615–641.

[126] Hashemi, S. H., Clarke, C. L., Kamps, J., Kiseleva, J., and Voorhees, E. M. (2016). Overview of the trec 2016 contextual suggestion track. In *TREC 2016*. NIST.

[127] Hashemi, S. H., Dehghani, M., and Kamps, J. (2015a). Parsimonious user and group profiling in venue recommendation. *Proceedings National Institute for Standards and Technology*, 500.

[128] Hashemi, S. H., Dehghani, M., and Kamps, J. (2015b). Parsimonious user and group profiling in venue recommendation. In *TREC 2015*. NIST.

[129] He, B. and Ounis, I. (2009a). Finding good feedback documents. In *CIKM '09*, pages 2011–2014.

[130] He, B. and Ounis, I. (2009b). Studying query expansion effectiveness. In *ECIR'09*, pages 611–619.

[131] Henaff, M., Weston, J., Szlam, A., Bordes, A., and LeCun, Y. (2016). Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*.

[132] Hensman, J., Matthews, A. G. d. G., and Ghahramani, Z. (2015). Scalable variational gaussian process classification. In *Proceedings of AISTATS*.

[133] Herbrich, R., Graepel, T., and Obermayer, K. (1999). Support vector learning for ordinal regression. In *ICANN '99*, pages 97–102.

[134] Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

[135] Hiemstra, D., Kamps, J., Kaptein, R., and LI, R. (2008). Parsimonious language models for a terabyte of text. In *The Sixteenth Text REtrieval Conference Proceedings*, TREC 2007. NIST.

[136] Hiemstra, D., Robertson, S., and Zaragoza, H. (2004). Parsimonious language models for information retrieval. In *SIGIR '04*, pages 178–185.

[137] Hinton, G., Vinyals, O., and Dean, J. (2014). Distilling the knowledge in a neural network. In *NIPS 2014 Deep Learning Workshop*. arXiv preprint arXiv:1503.02531.

[138] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554.

[139] Hirst, G., Riabinin, Y., Graham, J., and Boizot-Roche, M. (2014). Text to ideology or text to party status? *From Text to Political Positions: Text analysis across disciplines*, 55:93–15.

[140] Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J. (2003). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *A Field Guide to Dynamical Recurrent Neural Networks*.

[141] Hoffmann, R., Zhang, C., Ling, X., Zettlemoyer, L., and Weld, D. S. (2011). Knowledge-based weak supervision for information extraction of overlapping relations. In *HLT '11*, pages 541–550.

[142] Holyoak, K. J. (2012). Analogy and relational reasoning. *The Oxford handbook of thinking and reasoning*.

[143] Hu, L., Cao, J., Xu, G., Cao, L., Gu, Z., and Cao, W. (2014). Deep modeling of group preferences for group-based recommendation. In *AAAI*.

[144] Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., and Heck, L. (2013). Learning deep structured semantic models for web search using clickthrough data. In *CIKM '13*, pages 2333–2338.

[145] Hume, D. (2003). *A treatise of human nature*. Courier Corporation.

[146] Inhelder, B. and Piaget, J. (1958). *The growth of logical thinking from childhood to adolescence: An essay on the construction of formal operational structures*, volume 22. Psychology Press.

[147] Jain, S. and Wallace, B. C. (2014). Attention is not explanation. *CoRR*, abs/1902.10186.

[148] Jameson, A. and Smyth, B. (2007). The adaptive web. In Brusilovsky, P., Kobsa, A., and Nejdl, W., editors, *Recommendation to Groups*, pages 596–627. Springer-Verlag, Berlin, Heidelberg.

[149] Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

[150] Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM.

[151] Joulin, A. and Mikolov, T. (2015). Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in Neural Information Processing Systems*, NIPS'15.

[152] Kaiser, L., Gomez, A. N., and Chollet, F. (2017). Depthwise separable convolutions for neural machine translation. *CoRR*, abs/1706.03059.

[153] Kaiser, L., Gomez, A. N., and Chollet, F. (2018). Depthwise separable convolutions for neural machine translation. In *International Conference on Learning Representations*.

[154] Kaiser, L. and Sutskever, I. (2016). Neural GPUs learn algorithms. In *International Conference on Learning Representations*, ICLR'16.

[155] Kalchbrenner, N., Espeholt, L., Simonyan, K., van den Oord, A., Graves, A., and Kavukcuoglu, K. (2016). Neural machine translation in linear time. *CoRR*, abs/1610.10099.

[156] Kanoulas, E., Pavlu, V., Dai, K., and Aslam, J. (2009). Modeling the score distributions of relevant and non-relevant documents. In *ICTIR'09*, volume 5766, pages 152–163. Springer Berlin Heidelberg.

[157] Kaptein, R., Kamps, J., and Hiemstra, D. (2009). The impact of positive, negative and topical relevance feedback. In *The Seventeenth Text REtrieval Conference Proceedings*, TREC 2008. NIST.

[158] Kenter, T., Borisov, A., Van Gysel, C., Dehghani, M., de Rijke, M., and Mitra, B. (2017). Neural networks for information retrieval. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[159] Kim, D.-k., Voelker, G., and Saul, L. K. (2013). A variational approximation for topic modeling of hierarchical corpora. In *ICML*, ICML'13, pages 55–63.

[160] Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Association for Computational Linguistics*.

[161] Kingma, D. and Ba, J. (2014a). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[162] Kingma, D. P. and Ba, J. (2014b). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

[163] Kiritchenko, S., Zhu, X., and Mohammad, S. M. (2014). Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762.

[164] Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

[165] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[166] Kumar, A., Irsoy, O., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., Zhong, V., Paulus, R., and Socher, R. (2016). Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

[167] Lafferty, J. and Zhai, C. (2001). Document language models, query models, and risk minimization for information retrieval. In *SIGIR '01*, pages 111–119.

[168] Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40.

[169] Lavrenko, V. and Croft, W. B. (2001). Relevance based language models. In *SIGIR '01*, pages 120–127.

[170] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[171] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

[172] Lee, D.-H. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, page 2.

[173] Lee, J., Lee, Y., Kim, J., Kosiorek, A. R., Choi, S., and Teh, Y. W. (2018). Set transformer. *arXiv preprint arXiv:1810.00825*.

[174] Lewis, D. D. (1992). *Representation and Learning in Information Retrieval*. PhD thesis, UMass Amherst, Amherst, MA, USA.

[175] Lewis, D. D. (1995). Evaluating and optimizing autonomous text classification systems. In *SIGIR '95*, pages 246–254.

[176] Lin, Y., Ji, H., Liu, Z., and Sun, M. (2018). Denoising distantly supervised open-domain question answering. In *56th Annual Meeting of the Association for Computational Linguistics*, pages 1736–1745.

[177] Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

[178] Linzen, T., Dupoux, E., and Goldberg, Y. (2016). Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association of Computational Linguistics*, 4(1):521–535.

[179] Liu, X., Gao, J., He, X., Deng, L., Duh, K., and Wang, Y.-y. (2015). Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *NAACL '15*, pages 912–921.

[180] Lopez-Paz, D., Bottou, L., Schölkopf, B., and Vapnik, V. (2016). Unifying distillation and privileged information. In *ICLR'16*. arXiv preprint arXiv:1511.03643.

[181] Losada, D. E. and Azzopardi, L. (2008). An analysis on document length retrieval trends in language modeling smoothing. *Inf. Retr.*, 11(2):109–138.

[182] Lu, Z. and Li, H. (2013). A deep architecture for matching short texts. In *NIPS '13*, pages 1367–1375.

[183] Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM J. Res. Dev.*, 2(2):159–165.

[184] Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025.

[185] Lv, Y. and Zhai, C. (2009a). Adaptive relevance feedback in information retrieval. In *CIKM '09*, pages 255–264.

[186] Lv, Y. and Zhai, C. (2009b). A comparative study of methods for estimating query language models with pseudo feedback. In *CIKM '09*, pages 1895–1898.

[187] Lv, Y. and Zhai, C. (2014). Revisiting the divergence minimization feedback model. In *CIKM '14*, pages 1863–1866.

[188] Macdonald, C. and Ounis, I. (2007). Expertise drift and query expansion in expert search. In *CIKM '07*, pages 341–350.

[189] Malach, E. and Shalev-Shwartz, S. (2017). Decoupling" when to update" from" how to update". In *NIPS2017*.

[190] Markie, P. (2004). Rationalism vs. empiricism.

[191] Marx, M. and Schuth, A. (2010). Dutchparl 1.0 a corpus of parliamentary documents in dutch. In *DIR*, pages 82–83.

[192] Masthoff, J. (2011). Group recommender systems: Combining individual models. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 677–702. Springer US.

[193] McCallum, A., Rosenfeld, R., Mitchell, T. M., and Ng, A. Y. (1998). Improving text classification by shrinkage in a hierarchy of classes. In *ICML*, pages 359–367.

[194] Meij, E., Weerkamp, W., Balog, K., and de Rijke, M. (2008). Parsimonious relevance models. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 817–818.

[195] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In *NIPS '13*, pages 3111–3119.

[196] Min, B., Grishman, R., Wan, L., Wang, C., and Gondek, D. (2013). Distant supervision for relation extraction with an incomplete knowledge base. In *HLT-NAACL*, pages 777–782.

[197] Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Association for Computational Linguistics*, pages 1003–1011.

[198] Mitchell, T. M. (1980). The need for biases in learning generalizations. Technical report, Department of Computer Science, Laboratory for Computer Science Research, Rutgers University.

[199] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., 1 edition.

[200] Mitra, B., Diaz, F., and Craswell, N. (2017a). Learning to match using local and distributed representations of text for web search. In *WWW '17*, pages 1291–1299.

[201] Mitra, B., Diaz, F., and Craswell, N. (2017b). Learning to match using local and distributed representations of text for web search. In *WWW '17*, pages 1291–1299.

[202] Miyato, T., Maeda, S.-i., Ishii, S., and Koyama, M. (2018). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*.

[203] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning*, ICML'10, pages 807–814.

[204] Nakov, P., Ritter, A., Rosenthal, S., Sebastiani, F., and Stoyanov, V. (2016). Semeval-2016 task 4: Sentiment analysis in twitter. *Proceedings of SemEval*, pages 1–18.

[205] Noroozi, M. and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer.

[206] Ogilvie, P. and Callan, J. (2004). Hierarchical language models for xml component retrieval. In *INEX*, pages 224–237.

[207] Oh, H.-S., Choi, Y., and Myaeng, S.-H. (2011). Text classification for a large-scale taxonomy using dynamically mixed local and global models for a node. In *ECIR*, pages 7–18.

[208] Olieman, A., Azarbonyad, H., Dehghani, M., Kamps, J., and Marx, M. (2014). Entity linking by focusing dbpedia candidate entities. In *Proceedings of the First International Workshop on Entity Recognition and Disambiguation*, ERD '14.

[209] Onal, K. D., Altingovde, I. S., Karagoz, P., and de Rijke, M. (2016). Getting started with neural models for semantic matching in web search. *arXiv preprint arXiv:1611.03305*.

[210] Ororbia II, A. G., Giles, C. L., and Reitter, D. (2015). Learning a deep hybrid model for semi-supervised text classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP'15.

[211] Paperno, D., Kruszewski, G., Lazaridou, A., Pham, N. Q., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., and Fernandez, R. (2016). The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1525–1534.

[212] Papernot, N., Abadi, M., Erlingsson, Ú., Goodfellow, I., and Talwar, K. (2017a). Semi-supervised knowledge transfer for deep learning from private training data. In *ICLR*. arXiv preprint arXiv:1610.05755.

[213] Papernot, N., Abadi, M., Erlingsson, Ú., Goodfellow, I., and Talwar, K. (2017b). Semi-supervised knowledge transfer for deep learning from private training data. In *ICLR*. arXiv preprint arXiv:1610.05755.

[214] Parikh, A., Täckström, O., Das, D., and Uszkoreit, J. (2016). A decomposable attention model. In *Empirical Methods in Natural Language Processing*.

[215] Pass, G., Chowdhury, A., and Torgeson, C. (2006). A picture of search. In *InfoScale '06*.

[216] Patrini, G., Nielsen, F., Nock, R., and Carioni, M. (2016). Loss factorization, weakly supervised learning and label noise robustness. In *International Conference on Machine Learning*, pages 708–717.

[217] Patrini, G., Rozza, A., Menon, A., Nock, R., and Qu, L. (2017). Making neural networks robust to label noise: a loss correction approach. In *CVPR*. arXiv preprint arXiv:1609.03683.

[218] Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global Vectors for Word Representation. In *EMNLP '14*, pages 1532–1543.

[219] Pepperberg, I. M. (2017). Animal language studies: What happened? *Psychonomic bulletin & review*, 24(1):181–185.

[220] Phan, N., Wang, Y., Wu, X., and Dou, D. (2016). Differential privacy preservation for deep auto-encoders: an application of human behavior prediction. In *AAAI*, pages 1309–1316.

[221] PoliticalMashup (2015). Political mashup project. *http://politicalmashup.nl/* and *http://schema.politicalmashup.nl/*. Netherlands Organization for Scientific Research.

[222] Quiroz, L., Mennes, L., Dehghani, M., Kanoulas, E., et al. (2016). Distributional semantics for medical information extraction. In *CLEF Working Notes*, pages 109–122.

[223] Rae, J., Hunt, J. J., Danihelka, I., Harley, T., Senior, A. W., Wayne, G., Graves, A., and Lillicrap, T. (2016). Scaling memory-augmented neural networks with sparse reads and writes. In *Advances in Neural Information Processing Systems*, pages 3621–3629.

[224] Raghunathan, A., Frostig, R., Duchi, J., and Liang, P. (2016). Estimation from indirect supervision with linear moments. In *International Conference on Machine Learning*, pages 2568–2577.

[225] Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. In *Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

[226] Ratner, A. J., De Sa, C. M., Wu, S., Selsam, D., and Ré, C. (2016). Data programming: Creating large training sets, quickly. In *Advances in Neural Information Processing Systems*, pages 3567–3575.

[227] Ravi, S. and Larochelle, H. (2016). Optimization as a model for few-shot learning. In *ICLR*.

[228] Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., and Rabinovich, A. (2015). Training deep neural networks on noisy labels with bootstrapping. In *ICLR2015-Workshop*.

[229] Rekatsinas, T., Chu, X., Ilyas, I. F., and Ré, C. (2017). Holoclean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(11):1190–1201.

[230] Riezler, S., Simianer, P., and Haas, C. (2014). Response-based learning for grounded machine translation. In *Association for Computational Linguistics*, pages 881–891.

[231] Robertson, S. (1977). The probability ranking principle in ir. *Journal of Documentation*, 33(4):294–304.

[232] Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

[233] Robertson, S. E. and Jones, K. S. (1976). Relevance weighting of search terms. *JASIS*, 27(3):129–146.

[234] Rocchio, J. (1971). *Relevance Feedback in Information Retrieval*, pages 313–323. Prentice-Hall, Inc. reprinted from Scientific Report ISR-9, Computation Laboratory, Harvard University, August 1965.

[235] Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2014). Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.

[236] Rosenberg, C., Hebert, M., and Schneiderman, H. (2005). Semi-supervised self-training of object detection models. In *Seventh IEEE Workshop on Applications of Computer Vision*.

[237] Rosenthal, S., Farra, N., and Nakov, P. (2017). Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval'17, pages 502–518.

[238] Rosenthal, S., Nakov, P., Kiritchenko, S., Mohammad, S. M., Ritter, A., and Stoyanov, V. (2015). Semeval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th international workshop on semantic evaluation*, SemEval'15, pages 451–463.

[239] Roth, D. (2017). Incidental supervision: Moving beyond supervised learning. In *AAAI*, pages 4885–4890.

[240] Rouvier, M. and Favre, B. (2016). Sensei-lif at semeval-2016 task 4: Polarity embedding fusion for robust sentiment analysis. *Proceedings of SemEval*, pages 202–208.

[241] Ruder, S. (2019). *Neural Transfer Learning for Natural Language Processing*. PhD thesis, National University of Ireland Galway.

[242] Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

[243] Ruthven, I. and Lalmas, M. (2003). A survey on the use of relevance feedback for information access systems. *Knowl. Eng. Rev.*, 18(2):95–145.

[244] Salton, G. and Yang, C.-S. (1973). On the specification of term values in automatic indexing. *Journal of documentation*, 29(4):351–372.

[245] Saracevic, T. (1975). Relevance: A review of the literature and a framework for thinking on the notion in information science. *JASIST*, 26:321–343.

[246] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47.

[247] Senot, C., Kostadinov, D., Bouzid, M., Picault, J., and Aghasaryan, A. (2011). Evaluation of group profiling strategies. In *IJCAI*, pages 2728–2733.

[248] Seo, M., Kembhavi, A., Farhadi, A., and Hajishirzi, H. (2017). Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*.

[249] Seo, M., Min, S., Farhadi, A., and Hajishirzi, H. (2016). Query-reduction networks for question answering. *arXiv preprint arXiv:1606.04582*.

[250] Severyn, A. and Moschitti, A. (2015a). Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 959–962. ACM.

[251] Severyn, A. and Moschitti, A. (2015b). Unitn: Training deep convolutional neural network for twitter sentiment classification. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Association for Computational Linguistics, Denver, Colorado*, pages 464–469.

[252] Shang, S., Hui, Y., Hui, P., Cuff, P., and Kulkarni, S. (2014). Beyond personalization and anonymity: Towards a group-based recommender system. In *SAC*, pages 266–273.

[253] Shen, Y., He, X., Gao, J., Deng, L., and Mesnil, G. (2014). Learning semantic representations using convolutional neural networks for web search. In *WWW '14*, pages 373–374.

[254] Shen, Y., Huang, P.-S., Gao, J., and Chen, W. (2017). Reasonet: Learning to stop reading in machine comprehension. In *23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055.

[255] Shen, Y., Seeger, M., and Ng, A. Y. (2006). Fast gaussian process regression using kd-trees. In *Advances in neural information processing systems*, pages 1225–1232.

[256] Shokri, R. and Shmatikov, V. (2015). Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321. ACM.

[257] Shokri, R., Stronati, M., and Shmatikov, V. (2016). Membership inference attacks against machine learning models. *arXiv preprint arXiv:1610.05820*.

[258] Sigurbjörnsson, B., Kamps, J., and de Rijke, M. (2004). An element-based approach to xml retrieval. In *INEX*, pages 19–26.

[259] Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087.

[260] Socher, R., Ganjoo, M., Manning, C. D., and Ng, A. (2013). Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943.

[261] Sodian, B. and Wimmer, H. (1987). Children's understanding of inference as a source of knowledge. *Child development*, pages 424–433.

[262] Song, Y. and Roth, D. (2014). On dataless hierarchical text classification. In *AAAI*, pages 1579–1585.

[263] Sparck Jones, K., Robertson, S., Hiemstra, D., and Hugo, Z. (2003). Language modeling and relevance. In *Language Modeling for Information Retrieval*, pages 57–71.

[264] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

[265] Stewart, R. and Ermon, S. (2017). Label-free supervision of neural networks with physics and domain knowledge. In *AAAI*, pages 2576–2582.

[266] Sukhbaatar, S., Bruna, J., Paluri, M., Bourdev, L., and Fergus, R. (2015a). Training convolutional networks with noisy labels. In *Workshop contribution at ICLR 2015.*

[267] Sukhbaatar, S., szlam, a., Weston, J., and Fergus, R. (2015b). End-to-end memory networks. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc.

[268] Sun, A. and Lim, E.-P. (2001). Hierarchical text classification and evaluation. In *ICDM*, pages 521–528.

[269] Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. *arXiv preprint arXiv:1707.02968*.

[270] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

[271] Tabrizi, S. A., Dadashkarimi, J., Dehghani, M., Nasr Esfahani, H., and Shakery, A. (2015). Revisiting optimal rank aggregation: A dynamic programming approach. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, ICTIR '15.

[272] Tang, L., Wang, X., and Liu, H. (2011). Group profiling for understanding social structures. *TOIS*, 3(1):15:1–15:25.

[273] Tang, Y. (2016). Tf.learn: Tensorflow's high-level module for distributed machine learning. *arXiv preprint arXiv:1612.04251*.

[274] Tao, T. and Zhai, C. (2006). Regularized estimation of mixture models for robust pseudo-relevance feedback. In *SIGIR '06*, pages 162–169.

[275] Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. (2006). Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

[276] Tenenbaum, J. B., Kemp, C., Griffiths, T. L., and Goodman, N. D. (2011). How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285.

[277] Terra, E. and Warren, R. (2005). Poison pills: Harmful relevant documents in feedback. In *CIKM '05*, pages 319–320.

[278] Timpf, S. (1999). Abstraction, levels of detail, and hierarchies in map series. In *International conference on spatial information theory*, pages 125–139. Springer.

[279] Titsias, M. K. (2009). Variational learning of inducing variables in sparse gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 567–574.

[280] Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., and Ristenpart, T. (2016). Stealing machine learning models via prediction apis. In *USENIX Security*.

[281] Tran, K., Bisazza, A., and Monz, C. (2018a). The importance of being recurrent for modeling hierarchical structure. In *Proceedings of NAACL'18*.

[282] Tran, K., Bisazza, A., and Monz, C. (2018b). The importance of being recurrent for modeling hierarchical structure. In *Proceedings of NAACL'18*.

[283] Trask, A., Hill, F., Reed, S. E., Rae, J., Dyer, C., and Blunsom, P. (2018). Neural arithmetic logic units. In *Advances in Neural Information Processing Systems*, pages 8046–8055.

[284] Vahdat, A. (2017). Toward robustness against label noise in training deep discriminative neural networks. In *NIPS '17*.

[285] Van Rijsbergen, C., Harper, D., and Porter, M. (1981). The selection of good search terms. *IP&M*, 17:77–91.

[286] Van Rijsbergen, C. J. (1986). A new theoretical framework for information retrieval. *SIGIR Forum*, 21(1-2):23–29.

[287] Vapnik, V. and Izmailov, R. (2015). Learning using privileged information: similarity control and knowledge transfer. *Journal of machine learning research*, 16(20232049):55.

[288] Vapnik, V. and Vashist, A. (2009). A new learning paradigm: Learning using privileged information. *Neural networks*, 22(5):544–557.

[289] Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley-Interscience.

[290] Varma, P., He, B., Iter, D., Xu, P., Yu, R., De Sa, C., and Ré, C. (2017). Socratic learning: Correcting misspecified generative models using discriminative models. *arXiv preprint arXiv:1610.08123*.

[291] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017a). Attention is all you need. *CoRR*.

[292] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017b). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

[293] Veit, A., Alldrin, N., Chechik, G., Krasin, I., Gupta, A., and Belongie, S. (2017). Learning from noisy large-scale datasets with minimal supervision. In *The Conference on Computer Vision and Pattern Recognition*.

[294] Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. (2016). Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638.

[295] Voorhees, E. M. (2003). Overview of the trec 2003 robust retrieval track. In *TREC 2003*, pages 69–77.

[296] Wainwright, M. J., Jordan, M. I., and Duchi, J. C. (2012). Privacy aware learning. In *Advances in Neural Information Processing Systems*, pages 1430–1438.

[297] Wang, S., Yu, M., Guo, X., Wang, Z., Klinger, T., Zhang, W., Chang, S., Tesauro, G., Zhou, B., and Jiang, J. (2018a). R3: Reinforced reader-ranker for open-domain question answering. In *The Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5981–5988.

[298] Wang, S., Yu, M., Jiang, J., Zhang, W., Guo, X., Chang, S., Wang, Z., Klinger, T., Tesauro, G., and Campbell, M. (2018b). Evidence aggregation for answer re-ranking in open-domain question answering. In *International Conference on Learning Representations*.

[299] Warren, R. H. and Liu, T. (2004). A review of relevance feedback experiments at the 2003 reliable information access (ria) workshop. In *SIGIR '04*, pages 570–571.

[300] Wauthier, F. L., Jordan, M. I., and Jojic, N. (2013). Efficient ranking from pairwise comparisons. In *ICML'13*, pages 109–117.

[301] Welling, M., Rosen-Zvi, M., and Hinton, G. E. (2005). Exponential family harmoniums with an application to information retrieval. In *Advances in neural information processing systems*, pages 1481–1488.

[302] Weston, J., Bordes, A., Chopra, S., Rush, A. M., van Merriënboer, B., Joulin, A., and Mikolov, T. (2015). Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

[303] Weston, J., Ratle, F., Mobahi, H., and Collobert, R. (2012). Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer.

[304] Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

[305] Wilson, A. G. and Nickisch, H. (2015). Kernel interpolation for scalable structured gaussian processes. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1775–1784.

[306] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural

machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

[307] Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. (2019). Unsupervised data augmentation. *arXiv preprint arXiv:1904.12848*.

[308] Xiong, C., Dai, Z., Callan, J., Liu, Z., and Power, R. (2017a). End-to-end neural ad-hoc ranking with kernel pooling. In *SIGIR '17*, pages 55–64.

[309] Xiong, C., Zhong, V., and Socher, R. (2017b). Dynamic coattention networks for question answering. In *International Conference on Learning Representations*.

[310] Xue, G.-R., Dai, W., Yang, Q., and Yu, Y. (2008a). Topic-bridged plsa for cross-domain text classification. In *SIGIR '08*, pages 627–634.

[311] Xue, G.-R., Xing, D., Yang, Q., and Yu, Y. (2008b). Deep classification in large-scale text hierarchies. In *SIGIR*, pages 619–626.

[312] Yang, G. H. and Zhang, S. (2017). Differential privacy for information retrieval. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, pages 325–326.

[313] Yang, L., Ai, Q., Guo, J., and Croft, W. B. (2016). anmm: Ranking short answer texts with attention-based neural matching model. In *CIKM '16*, pages 287–296.

[314] Yang, W., Lu, K., Yang, P., and Lin, J. (2019). Critically examining the "neural hype": Weak baselines and the additivity of effectiveness gains from neural ranking models. *arXiv preprint arXiv:1904.09171*.

[315] Yao, L., Mimno, D., and McCallum, A. (2009). Efficient methods for topic model inference on streaming document collections. In *SIGKDD*, pages 937–946.

[316] Yogatama, D., Miao, Y., Melis, G., Ling, W., Kuncoro, A., Dyer, C., and Blunsom, P. (2018). Memory architectures in recurrent neural network language models. In *International Conference on Learning Representations*.

[317] Yu, B., Kaufmann, S., and Diermeier, D. (2008). Classifying party affiliation from political speech. *Journal of Information Technology & Politics*, 5(1):33–48.

[318] Yu, Z., Zhou, X., Hao, Y., and Gu, J. (2006). Tv program recommendation for multiple viewers based on user profile merging. *User Modeling and User-Adapted Interaction*, 16(1):63–82.

[319] Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). Deep sets. In *Advances in neural information processing systems*, pages 3391–3401.

[320] Zamani, H. and Croft, W. B. (2016a). Embedding-based query language models. In *ICTIR '16*, pages 147–156.

[321] Zamani, H. and Croft, W. B. (2016b). Estimating embedding vectors for queries. In *ICTIR '16*, pages 123–132.

[322] Zamani, H. and Croft, W. B. (2017). Relevance-based word embedding. In *SIGIR '17*.

[323] Zamani, H. and Croft, W. B. (2018). On the theory of weak supervision for information retrieval. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '18, pages 147–154.

[324] Zamani, H., Dehghani, M., Croft, W. B., Learned-Miller, E., and Kamps, J. (2018a). From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18.

[325] Zamani, H., Dehghani, M., Diaz, F., Li, H., and Craswell, N. (2018b). Workshop on learning from limited or noisy data for information retrieval. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '18.

[326] Zaremba, W. and Sutskever, I. (2015). Learning to execute. *CoRR*, abs/1410.4615.

[327] Zavitsanos, E., Paliouras, G., and Vouros, G. A. (2011). Non-parametric estimation of topic hierarchies from texts with hierarchical dirichlet processes. *J. Mach. Learn. Res.*, 12:2749–2775.

[328] Zhai, C. (2008). Statistical language models for information retrieval. *Synthesis Lectures on Human Language Technologies*, 1(1):1–141.

[329] Zhai, C. and Lafferty, J. (2001a). Model-based feedback in the language modeling approach to information retrieval. In *CIKM '01*, pages 403–410.

[330] Zhai, C. and Lafferty, J. (2001b). A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR '01*, pages 334–342.

[331] Zhang, Y., Rahman, M. M., Braylan, A., Dang, B., Chang, H.-L., Kim, H., McNamara, Q., Angert, A., Banner, E., Khetan, V., et al. (2016). Neural information retrieval: A literature review. *arXiv preprint arXiv:1611.06792*.

[332] Zheng, G. and Callan, J. (2015). Learning to Reweight Terms with Distributed Representations. In *SIGIR '15*, pages 575–584.

[333] Zhou, D., Xiao, L., and Wu, M. (2011). Hierarchical classification via orthogonal transfer. In *ICML*, pages 801–808.

[334] Zhou, Z.-H. (2018). A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53.

Summary

Learning with Imperfect Supervision for Language Understanding

Humans learn to solve complex problems and uncover underlying concepts and relations given limited, noisy or inconsistent observations and draw successful generalizations based on them. This rests largely on the *poverty of the stimulus* argument, or what is sometimes called *Plato's problem*: "How do we know so much when the evidence available to us is so meagre?"

In contrast, the success of today's data-driven machine learning models is often strongly correlated with the amount of available high quality labeled data and teaching machines using *imperfect supervision* remains a key challenge. In practice, however, for many applications, large-scaled high-quality training data is not available, which highlights the increasing need for building models with the ability to learn complex tasks with imperfect supervision, i.e., where the learning process is based on imperfect training samples.

When designing learning algorithms, pure data-driven learning, which relies only on previous experience, does not seem to be able to learn generalizable solutions. Similar to human's innately primed learning, having part of the knowledge encoded in the learning algorithms in the form of strong or weak biases, can help learning solutions that better generalize to unseen samples.

In this thesis, we focus on the problem of **the poverty of stimulus for learning algorithms**. We argue that even noisy and limited signals can contain a great deal of valid information that can be incorporated along with prior knowledge and biases that are encoded into learning algorithms in order to solve complex problems. We improve the process of learning with imperfect supervision by (i) *employing prior knowledge in learning algorithms*, (ii) *augmenting data and learning to learn how to better use the data*, and (iii) *introducing inductive biases to learning algorithms* . These general ideas are, in fact, the key ingredients for building any learning algorithms that can generalize beyond (imperfections in) the observed data.

We concentrate on language understanding and reasoning, as one of the extraordinary cognitive abilities of humans, as well as a pivotal problem in artificial intelligence. We try to improve the learning process, in more principled ways than ad-hoc and domain or task-specific tricks to improve the output. We investigate our ideas on a wide range of sequence modeling and language understanding tasks.

Samenvatting

Leren met Imperfecte Supervisie om Taal te Begrijpen

Mensen leren complexe problemen op te lossen en onderliggende concepten en relaties te ontdekken op basis van beperkte, ruizige of inconsistente observaties en kunnen daarmee succesvol generaliseren. Dit berust grotendeels op het argument van *de armoede van de stimulus*, ofwel *het probleem van Plato*: "Hoe weten we zoveel wanneer het beschikbare bewijs zo mager is?"

Daarentegen is het succes van machine leren vaak sterk gecorreleerd met de hoeveelheid beschikbare hoge kwaliteit, gelabelde data. Machine leren met beperkt gelabelde data of met *imperfecte supervisie* blijft een belangrijke uitdaging. In de praktijk is echter vaak geen grote hoeveelheid gelabelde data van hoge kwaliteit beschikbaar. Hierdoor ontstaat een toenemende behoefte om modellen te ontwerpen die complexe taken met imperfecte supervisie kunnen oplossen, oftewel waarbij het leerproces op imperfect gelabelde data is gebaseerd.

Bij het ontwerpen van algoritmen lijken methodes die enkel op basis van eerdere ervaringen leren geen generaliserende oplossingen op te leveren. Net als bij menselijk leren, waarbij een deel van de kennis van nature aanwezig is, kunnen ook leeralgoritmen mogelijk beter generaliseren wanneer een deel van de kennis gecodeerd wordt in het algoritme in de vorm van voorkennis.

In dit proefschrift richten we ons op **het probleem van de armoede van de stimulus voor machine leren**. Wij stellen dat zelfs beperkte, ruizige signalen nuttige informatie kunnen bevatten. Zulke signalen kunnen, samen met voorkennis die wordt gecodeerd in de algoritmen, gebruikt worden om complexe problemen op te lossen. We verbeteren het leerproces met imperfecte supervisie door (i) *voorkennis in leeralgoritmen toe te passen*, (ii) *data kunstmatig aan te vullen en te leren hoe de data beter te gebruiken is*, en (iii) *inductieve bias in de leeralgoritmen te introduceren*. Deze algemene ideeën zijn in feite de belangrijkste ingrediënten voor het bouwen van leeralgoritmen die beter kunnen generaliseren bij beperkt gelabelde data.

We concentreren ons op het begrijpen van en redeneren met taal, één van de unieke cognitieve vaardigheden van de mens en een cruciaal probleem in de kunstmatige intelligentie. We proberen het leerproces te verbeteren op principiële manieren die ad-hoc en domein- of taakspecifieke trucs vermijden. We onderzoeken onze ideeën aan de hand van een breed scala van taken zoals het modelleren van sequenties en taalbegrip.