Simple Transformers: Open-source for All

Thilina Chaturanga Rajapakse
University of Amsterdam
The Netherlands
t.c.r.rajapakse@uva.nl

Andrew Yates
University of Amsterdam
The Netherlands
a.c.yates@uva.nl

Maarten de Rijke
University of Amsterdam
The Netherlands
m.derijke@uva.nl

Abstract

Language technology, particularly information retrieval, is poised to have a profound impact on society. We believe that technology with such far-reaching potential should be accessible to everyone, not just the technologically privileged. Therefore, we advocate for **open-source for all**, ensuring that individuals from diverse research areas, societal sectors, and backgrounds have access to information retrieval and language technology tools with low barriers to entry. In this paper, we describe Simple Transformers, a library created with these goals in mind. It is designed to simplify the training, evaluation, and usage of transformer models. As of 2024, the library has garnered over 4,000 stars on GitHub and has been downloaded over 3 million times. These metrics reflect its wide acceptance and usage across different sectors. We describe the design and implementation of the library, provide examples of its usage and adoption, Finally, we also reflect on how Simple Transformers contributes to the goal of "open-source for all."

CCS Concepts

• **Information systems** → **Information retrieval**.

Keywords

Open source information retrieval

ACM Reference Format:
Thilina Chaturanga Rajapakse, Andrew Yates, and Maarten de Rijke. 2024. Simple Transformers: Open-source for All. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region (SIGIR-AP '24), December 9–12, 2024, Tokyo, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3673791.3698412

1 Introduction

Many of us see the goal of research as the improvement of people's lives everywhere. As researchers, we have the opportunity to make new discoveries and create innovations that help advances in dealing with challenges in climate change, education, healthcare, media, mobility, culture, government, and business [34] – language technology in general, and information retrieval in particular, is likely to impact all of these areas. In an open society, technology with such a wide and pervasive reach should be in the hands of many, not of a few [38]. For us, this point of view entails three things: (i) a focus on shared innovation, where the knowledge and technology being created and examined is shared and owned by multiple, and multiple types of, stakeholders – academic, industrial, governmental, and societal [23]; (ii) a focus on empowering people, equipping them with new tools, rather than replacing them [1, 35]; and (iii) a focus on getting many voices around the table, as the technology is likely to affect many [5, 37].

In our view, open-source software is key in each of those three dimensions. Clearly, open-source software feeds and is fed by shared innovation ecosystems, thereby fostering collaboration [36]. By fostering accessibility, open-source software empowers people, not just by code, data, and tool sharing but also by reducing the need for repeated large-scale investments [12]. Finally, open-source software drives transparency, which helps to build trust, which in turn helps to increase engagement.

None of the points that we have made so far is new. The open-source community has made impressive, valuable contributions towards advancing the goal of shared innovation (i). For example, in language technology, the BLOOM initiative [33] organized by the BigScience-community produced the first multilingual large language model (LLM) trained in complete transparency as a result of the largest collaboration of AI researchers ever involved in a single research project [33]. And in information retrieval, Lucene, the open-source search engine software library, has been in active usage for over 25 years [10].

In this paper, we draw attention to the second focus we outlined: empowering people by equipping them with new (information retrieval and language technology) tools, not just specialists and people who focus on language technology. We hope that by providing people with accessible tools to experiment with and use information retrieval and language technology, we can also advance on the third focus, i.e., getting many voices around the table to discuss and make decisions about information retrieval language technology that will impact the lives of many.

This paper describes one instance of this idea: **Simple Transformers**. Transformers [42] and transformer-based architectures have found usage across, and far beyond, information retrieval and language technology. Simple Transformers is an open-source library for training, evaluating, and using transformer models. The core design philosophy for the library is that using these transformative technologies should not be restricted to only those with expertise in the field. Instead, the library is designed to be accessible to and used by as broad a community as possible.

Simple Transformers has a special focus on information retrieval (IR) techniques and other use cases of transformer models. It provides an easy interface to train, test, and use neural IR methods such as dense retrieval. For example, building a dense vector index of a document collection (using Faiss [8, 14]) is encapsulated to reduce complexity for the user. We demonstrate how to train a dual-encoder dense retrieval model [16] in Section 4.3.1.

This work is licensed under a Creative Commons Attribution International 4.0 License.

SIGIR-AP '24, December 9–12, 2024, Tokyo, Japan
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0724-7/24/12
https://doi.org/10.1145/3673791.3698412

The remainder of this paper focuses on the Simple Transformers library. We introduce the library in Section 3, elaborate on the design, and provide code examples in Section 4. In Section 6, we look at instances where Simple Transformers has been used outside the core information retrieval and language technology communities, demonstrating our contributions towards empowering people with new language technology tools. Finally, we discuss some limitations and our commitments for future work and reflect on our ambition of "open-source for all" in Section 7.

2 Related Work

The IR community has a long tradition of foundational open-source. In addition to Apache Lucene (mentioned in the introduction), prominent examples include the Terrier IR platform that implements state-of-the-art indexing and retrieval functionalities, with a focus on the rapid development and evaluation of large-scale retrieval applications [27]. Furthermore, Pyserini [20], Tevatron [9], the LSR library [24], and Sentence Transformers [31] are among popular libraries supporting the training and evaluation of IR methods and models.

These libraries are valuable tools widely used in the IR community. Our goal with Simple Transformers, however, is to reach a broader audience. As we describe in Section 6, and as evidenced by the popularity of the Hugging Face suite of libraries [11, 19, 43], NLP methods have been adopted in many fields beyond language technology. By integrating IR techniques alongside NLP methods with similar design principles and a familiar interface (as described in Section 4), we aim to make IR methods and techniques more accessible to users who may not possess expert knowledge of IR. While the Huggingface libraries support the *architectures* of the models used in IR, they do not provide methods of directly training or evaluating dense retrieval models, which can be a significant barrier to users unfamiliar with the (often complex) pipelines used in IR tasks. On the other hand, Simple Transformers provide accessible functions to directly implement IR pipelines as we discuss in Section 4.

3 Simple Transformers

Transformer models have become the most ubiquitous neural network architecture for natural language processing, eclipsing previous architectures like convolutional neural networks (CNNs) and recurrent neural networks (RNNs) [43]. The typical process for using a transformer model on a new task involves identifying a suitable pre-trained model (e.g., BERT, RoBERTa, BART), preprocessing the training data and converting the text into input tokens that can be processed by a transformer (tokenization), training the transformer on new data starting from the pre-trained weights (finetuning), and finally making predictions on similarly preprocessed test data.

While transformers can be trained to perform exceedingly well on many IR and NLP tasks, the finetuning process can be complicated depending on the specific models used and the task at hand. The Simple Transformers library was created to simplify this process and offer a streamlined and straightforward way to train, finetune, and perform predictions with many different transformer models on various IR and NLP tasks. Building on top of the Hugging Face Transformers library [43], thereby enabling access to the vast collection of Transformer models available on the Hugging Face Hub,[1] Simple Transformers provides an interface that is both easy to use and easy to understand (see Section 4.3 for examples).

As of 2024, the Simple Transformers library has accumulated over 4,000 stars on GitHub and has been downloaded over 3 million times, with an average of nearly 50,000 downloads per month according to PePy. Additionally, over 1,500 other GitHub repositories utilize the Simple Transformers library. These statistics bear testament to the usability and popularity of Simple Transformers, both within and without the core IR and NLP communities.

Simple Transformers is primarily designed for ease of use and accessibility for users who may not be familiar with IR, NLP or Deep Learning in general. We believe that this focus makes Simple Transformers a valuable resource for communities outside IR and NLP, such as researchers from other scientific disciplines and industry practitioners with broader software development skill sets, e.g., software engineers who are not specialized in IR and NLP.

The fields of IR and NLP encompass various tasks such as similarity search, text classification, question answering, and language generation. All these tasks and more are supported in the Simple Transformers library (see Table 1), with each task mapped to its own class for convenience. While the task-specific classes follow the same pipeline of training, evaluation, and prediction, there are necessary differences between different classes (e.g., data formats). The following section elaborates on this pipeline approach.

4 Design and Implementation

4.1 Setup

Simple Transformers is available as a Python package on PyPi[2] and can be installed through *pip*.

4.2 Design

Data. Simple Transformers defines input and output data formats according to each task. For example, the `RetrievalModel` class expects input data to contain the columns *query_text*, *gold_passage*, and an optional *title* column, while the `LanguageModelingModel` class expects the input data to contain a single column *text*. Data may be passed as a path to a file containing the required columns or as an in-memory Pandas dataframe. The specific input and output data formats for each task is defined in the documentation[3] of the Simple Transformers library.

Configuring Models. Each Simple Transformers class associated with a task has a configuration class (named after the task class, e.g., `ClassificationArgs`) that controls all hyperparameters and configuration options for that task. Crucially, the configuration class for each task comes with reasonable defaults that a novice user (a user unfamiliar with the task and/or model) can rely on to provide good results even without hyperparameter tuning. On the other hand, a more experienced user can perform hyperparameter tuning or otherwise change the configuration to improve the final

[1] https://huggingface.co/models
[2] https://pypi.org/project/simpletransformers/
[3] https://simpletransformers.ai/docs/installation/

Table 1: Overview of tasks and corresponding model classes in the Simple Transformers library.

Task	Model Class
Information retrieval (dense retrieval)	`RetrievalModel`
(Large) language models (training, fine-tuning, and generation)	`LanguageModelingModel`
Encoder model training and fine-tuning	`LanguageModelingModel`
Sequence classification	`ClassificationModel`
Token classification (NER)	`NERModel`
Question answering	`QuestionAnsweringModel`
Language generation	`LanguageGenerationModel`
Sequence to sequence (including Mono-T5 [26] style rerankers)	`T5Model`, `Seq2SeqModel`
Text representation	`RepresentationModel`

performance of the model. Again, the documentation of the library details the configuration options available for each task.

Training Models. Each Simple Transformers model class (task) has a `train_model()` method that accepts a dataset (in the correct input format) and initiates the training loop for the model. The hyperparameters for training can be configured when initializing the model class or alternatively can be passed to the `train_model()` function itself to update the initial configuration. Simple Transformers also has a built-in `evaluate_during_training` option to facilitate model training progress tracking by performing evaluation (optionally on a separate validation set) at set intervals as well as out-of-the-box early stopping functionality.

Evaluating Models. Similar to training, a model can be evaluated by calling the `eval_model()` function that accepts an evaluation dataset (in the correct input format). This method will compute a set of default metrics chosen to fit a particular task (e.g., classification models report *accuracy*, *F1*, etc., while retrieval models report *nDCG*, *recall*, etc.), but a user may also pass any additional metric functions that they wish to compute in addition to the default metrics. The `eval_model()` function also gets called internally when using the `evaluate_during_training` feature.

Predicting with Models. The `predict()` function of each model is called to make predictions on input data passed to the function. Unlike the `eval_model()` function, the `predict()` function does not compute metrics (and therefore does not require labels) and instead outputs the predictions from the model.

Visualization. The Simple Transformers library integrates two open-source visualization libraries, Weights & Biases[4] and Tensorboard.[5] These integrations can be used to track training and evaluation metrics easily. For example, simply setting the `wandb_project` option to a non-null value will automatically track training and evaluation metrics using the Weights & Biases library.

4.3 Examples

This section demonstrates two minimal examples of training and using transformer models with the Simple Transformers library. Example code for other models can be found in the Simple Transformers GitHub repository, and minimal start examples can be found in the documentation. Advanced configuration and techniques, such as custom parameter groups and hyperparameter tuning, are described in the documentation (and omitted from the paper for brevity).

4.3.1 Dense Retrieval. This example shows how to train a simple dense retrieval model like the dense passage retriever (DPR) [16] on the MSMARCO dataset [25]. The full code can be found in the Simple Transformers repository.[6]

Data Format. The format for the training data can be seen in Table 2. Note that the `hard_negative` column is optional, but using hard negatives typically yields better performance [44]. When the column is not present, in-batch negatives are used.

Training. To train a dense retrieval model with Simple Transformers, the `RetrievalArgs` configuration class is instantiated and the preferred hyperparameter values are set.

```
# Path to the training data
train_data_path = "msmarco-train.tsv"

model_args = RetrievalArgs()
model_args.use_hard_negatives = True
model_args.num_train_epochs = 40
model_args.train_batch_size = 16
model_args.learning_rate = 1e-6
```

Then, we define the pretrained model to be used for initializing the weights and initialize the `RetrievalModel` with the pretrained weights and the configured hyperparameter values.

```
model_type = "custom"
model_name = None
context_name = "bert-base-cased"
question_name = "bert-base-cased"

# Instantiate RetrievalModel
model = RetrievalModel(
    model_type,
    model_name,
    context_name,
    question_name,
    args=model_args,
)
```

The `train_model()` function is called as shown below to initiate the training of the dense retriever.

```
model.train_model(train_data)
```

[4] https://github.com/wandb/wandb
[5] https://github.com/tensorflow/tensorboard
[6] https://github.com/ThilinaRajapakse/simpletransformers/tree/master/examples/retrieval

Table 2: Data format to train a dense retrieval model.

query_text	gold_passage	hard_negative
what are the liberal arts?	liberal arts. 1. the academic course of instruction at a college intended to…	Liberal Education: An approach to college learning that empowers individuals…
what is the mechanism of action of fibrinolytic or thrombolytic drugs?	Baillière's Clinical Haematology. 6 Mechanism of action of the…	Be able to diagram the coagulation and fibrinolytic pathways and the…

Evaluation. Simple Transformers can be used to perform evaluation on datasets in TREC or BEIR [39] formats, as well as Pandas dataframes containing `query_text` and `gold_passage` columns. Simple Transformers automatically builds and stores a dense vector index, using Faiss, to perform retrieval.

When `save_as_experiment` is enabled, Simple Transformers will save per query metrics to facilitate easy statistical testing. Furthermore, the built-in `analyze_experiment()` function can be used to automatically perform statistical tests and generate latex table code for results saved with `save_as_experiment`.

```
model.eval_model(
    eval_data_path,
    save_as_experiment=True,
    pytrec_eval_metrics=[
        "recip_rank",
        "recall_100",
        "ndcg_cut_10",
    ],
)
```

4.3.2 Adapter Tuning a LLaMA model for retrieval augmented generation. This example demonstrates how to easily adapter tune [13] a LLaMA 7B[7] model [40] for retrieval augmented generation [18, 30]. Simple Transformers uses the Hugging Face PEFT [22], bitsandbytes, and accelerate [11] libraries to enable adapter tuning a LLaMA 7B model on consumer-grade GPUs (this example requires a GPU with ~12 GB of memory). Again, the full code can be found in the Simple Transformers repository.[8]

We use the pretrained DRAGON dense retriever model [21] and the MSMARCO collection as the dense retriever and document collection for retrieval, respectively.

Data Format. The data format for adapter tuning a large language model (LLM) is shown in Table 3. The `text` column contains the text that the language model will be (adapter) trained on, while the `rag_query` column contains the questions that will be passed to the retriever model. Note that the data format is identical for training or finetuning a language model without retrieval augmentation, except that the `rag_query` column is not required in this scenario.

Training. Following the same procedure as in Section 4.3.1, we instantiate the `LanguageModelingArgs` object and set the preferred hyperparameter values.

```
# Path to the training data
train_data_path = "train.jsonl"

model_args = LanguageModelingArgs()
model_args.peft = True
model_args.nf4 = True
model_args.loftq_bits = 4
model_args.lora_config = {"r": 8}
model_args.data_format = "jsonl"
model_args.optimizer = "Adam8bit"
model_args.max_seq_length = 500
```

Then, we define the LLM to adapt and the retriever we will use. Note that `retrieval_model` here has been instantiated similarly to the example in Section 4.3.1, and we omit its instantiation for brevity.

```
model_type = "causal"
model_name = "meta-llama/Llama-2-7b-hf"

model = LanguageModelingModel(
    "causal",
    "meta-llama/Llama-2-7b-hf",
    args=model_args,
    retrieval_model=retrieval_model,
)
```

Adapter tuning of the LLM is initiated by calling the `train_model()` function, again identical to the procedure followed to train the dense retrieval model earlier.

```
model.train_model(train_data)
```

Language Generation. The `predict()` function of the model class can be used to generate the LLM responses to an input, as shown below.

```
responses, _ = model.predict(
    to_predict,
    rag_queries=rag_queries,
)
```

Here, `to_predict` is a list of input prompts to the LLM, and the `rag_queries` is a list of queries used by the retriever to retrieve documents relevant to the question from the document collection. In practice, `rag_queries` are essentially the question itself without the prompt for the LLM (i.e., "*What is in front of the Notre Dame Main Building*" without the surrounding "*Question: …Answer:*" found in the LLM prompt).

5 Experiments

In this section, we replicate the results from DPR [16] by training and evaluating a DPR dense retrieval model[9] [10] using the Simple Transformers library. Following [16], we train the model on the Natural Questions (NQ) [17] dataset and evaluate on NQ, TriviaQA [15], WebQuestions (WQ) [4], CuratedTREC (TREC) [3], and

[7] https://huggingface.co/meta-llama/Llama-2-7b-hf
[8] https://github.com/ThilinaRajapakse/simpletransformers/tree/master/examples/llms

[9] https://huggingface.co/thilina/dpr-nq-st-query-encoder
[10] https://huggingface.co/thilina/dpr-nq-st-context-encoder

Table 3: Data format to adapter tune a retrieval-augmented large language model (LLM).

text	rag_query
Question: To whom did the Virgin Mary allegedly appear in 1858 in Lourdes France? Answer: Saint Bernadette Soubirous	To whom did the Virgin Mary allegedly appear in 1858 in Lourdes France?
Question: What is in front of the Notre Dame Main Building? Answer: a copper statue of Christ	What is in front of the Notre Dame Main Building?

SQuAD [29]. Following Karpukhin et al. [16], we train the model using a learning rate of $1e^{-5}$, and a batch size of 128 for 40 epochs, without early stopping.

Table 4: Top-20/Top-100 accuracy on the five evaluation datasets used in DPR [16].

Framework	NQ	TriviaQA	WQ	TREC	SQuAD
DPR [16]	78.4/85.4	79.4/85.0	73.2/81.4	79.8/89.1	63.2/77.2
Tevatron	79.8/86.9	80.2/85.5	75.4/82.9	84.0/90.7	62.3/77.0
ST	76.9/85.6	79.5/85.2	72.2/80.7	84.2/91.6	61.9/77.1

Table 4 shows the top-20 and top-100 accuracy metrics obtained by DPR models trained using the original DPR repository [16], the Tevatron library [9], and the Simple Transformers library. Overall, the top-k metrics are comparable for all three frameworks, indicating that the results were replicated successfully.

6 Adoption

This section looks at the adoption of Simple Transformers in research areas other than information retrieval and NLP. We believe this to be an important benefit of open-source libraries and technologies.

Materials Science. Cruse et al. [7] use transformer models via the Simple Transformers library to build a publicly available dataset of codified gold nanoparticle synthesis protocols and outcomes extracted from nearly 5 million existing nanoparticle materials science publications. They suggest that this data could help understand the underlying mechanisms controlling the size and shape of gold nanoparticles.

Environmental Social Sciences. Coan et al. [6] study the role of misinformation in shaping the debate on climate change. They used Simple Transformers to build a model to detect specific contrarian claims regarding climate change, as opposed to broad topics or themes [6]. They specifically note that interdisciplinary approaches are required to combat online misinformation efforts at the required scale. We believe that this highlights the need for accessible open-source tools that could be employed by researchers less familiar with IR and NLP tools and methods.

Healthcare. Owen et al. [28] observe that key symptoms of major depressive disorder can manifest in the structure of written language and that early diagnosis is critical for achieving accurate diagnosis and improving patient outcomes. Therefore, they find that automatic analysis of rich and regular written communications, such as social media or web forums, may provide opportunities for early intervention. They employ various transformer models, through Simple Transformers, for the purpose of text analysis to detect symptoms of major depressive disorder.

Economics. Trust et al. [41] study the potential of NLP and machine learning methods to understand how decision-making in society is influenced by news. They propose a transformer-based method using weak supervision for the identification of news articles about economic uncertainty and adapt to the calculation of the economic policy uncertainty (EPU) index [2] to their proposed strategy. The transformer-based method proposed by Trust et al. [41] yields significant improvement over the existing keyword search based method. Simple Transformers was used to train the transformer models.

Human Resources. Robson et al. [32] explore the use of AI in bridging the gap between workforce reskilling demands and the offerings of local college training programs. Their project, SkillSync™, leverages AI technologies, including a variation of the Siamese Multidepth Transformer-based Hierarchical Encoder (SMITH) [45] and other natural language understanding methods, to facilitate the alignment of job descriptions and course information with skills taxonomies.

7 Limitations, Future Work, and Reflections

7.1 Limitations

The primary limitation of the Simple Transformers library is that we do not focus on the deployment of transformer models once they are trained and tested. While this is less of a concern for researchers, industry users need to deploy, scale, and maintain transformer models in production. The `predict()` function of Simple Transformers models can be used to perform ad hoc predictions but integrating this with, e.g., web servers is beyond the scope of the library. However, any model trained with Simple Transformers is fully compatible with any tools that work with Hugging Face models.

Furthermore, Simple Transformers does not support the full range of models available on Hugging Face, but instead focuses on the most commonly used tasks such as information retrieval, classification, and language generation. These tasks are also commonly found in other fields of research, and we believe that the ease of use for these core tasks is crucial to democratize access to powerful IR and NLP capabilities without overwhelming users with the complexity required to manage and deploy a wider array of models. By streamlining the process for these key IR and NLP tasks, the Simple Transformers library aims to lower the barrier to entry, enabling more researchers and developers to leverage the power of transformer models. We believe that both general-purpose, large-scale tools like the Hugging Face suite of libraries and more specialized, beginner-friendly tools such as Simple Transformers play a crucial role in driving shared innovation. These tools empower a diverse

community to benefit from open-source language technology, and their continued popularity highlights their impact and importance.

While this focus limits the library's scope in terms of model variety and deployment capabilities, it ensures that users can quickly and effectively train, test, and use transformer models without requiring extensive expertise in IR and NLP. This approach allows the Simple Transformers library to maintain its simplicity and accessibility while offering users pathways to deploy their models in real-world applications through full compatibility with Hugging Face libraries.

7.2 Future Work

We aim to keep the Simple Transformers library up to date with the latest developments in information retrieval and natural language processing and to democratize access to the latest models and methodologies to a broader audience beyond the deep learning community.

7.3 Reflections on "Open-source for All"

We reflect on the three key focuses we outlined in Section 1 and how the Simple Transformers library contributes towards them.

(1) **Shared innovation:** We believe that most of the progress by the open-source community has been made towards this focus. The pace of advances in IR and NLP over the past few years bears testament to this fact. Simple Transformers is part of a large and growing movement, based on open-source developments and open science, dedicated to shared innovation in both industry and academia. However, we believe that there is still ample room to involve governmental and societal stakeholders, sectors where IR and language technology are less prevalent and even less understood, in this focus.
(2) **Empowering people:** The adoption of the Simple Transformers library, particularly outside the IR and NLP communities, and the research being conducted in other fields powered by IR and NLP technologies is evidence of progress towards the focus of empowering people with new tools. We believe that further progress can be made by increasing the availability of accessible tools, as well as increasing access to training materials and educational resources.
(3) **Diversity:** While it is difficult to directly estimate the impact of Simple Transformers on our third focus, we believe that bringing more people and diverse opinions to the discussion starts with empowering people to use language technology in the first place. Therefore, we hope that by pushing towards the second focus, we indirectly move towards the third focus as well.

Acknowledgments

This research was (partially) supported by the Dreams Lab, a collaboration between Huawei Finland, the University of Amsterdam, and the Vrije Universiteit Amsterdam, by the Dutch Research Council (NWO), under project numbers VI.Vidi.223.166, 024.004.022, NWA.1389.20.183, and KICH3.LTP.20.006, and by the European Union's Horizon Europe program under grant agreement No. 101070212. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

[1] Zeynep Akata, Dan Balliet, Maarten de Rijke, Frank Dignum, Virginia Dignum, Guszti Eiben, Antske Fokkens, Davide Grossi, Koen Hindriks, Holger Hoos, Haley Hung, Catholijn Jonker, Christof Monz, Mark Neerincx, Frans Oliehoek, Henri Prakken, Stefan Schlobach, Linda van der Gaag, Frank van Harmelen, Herke van Hoof, Birna van Riemsdijk, Aimee van Wynsberghe, Rineke Verbrugge, Bart Verheij, Piek Vossen, and Max Welling. 2020. A Research Agenda for Hybrid Intelligence: Augmenting Human Intellect With Collaborative, Adaptive, Responsible, and Explainable Artificial Intelligence. *Computer* 53, 08 (August 2020), 18–28.
[2] Scott R Baker, Nicholas Bloom, and Steven J Davis. 2016. Measuring Economic Policy Uncertainty. *The Quarterly Journal of Economics* 131, 4 (2016), 1593–1636.
[3] Petr Baudiš and Jan Šedivỳ. 2015. Modeling of the Question Answering Task in the YodaQA system. In *International Conference of the Cross-language Evaluation Forum for European Languages*. Springer, 222–228.
[4] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 1533–1544.
[5] Claudi L. Bockting, Eva A. M. van Dis, Robert van Rooij, Willem Zuidema, and Johan Bollen. 2023. Living Guidelines for Generative AI — Why Scientists Must Oversee Its Use. *Nature* 622 (2023), 693–696.
[6] Travis G. Coan, Constantine Boussalis, John Cook, and Mirjam O. Nanko. 2021. Computer-assisted Classification of Contrarian Claims about Climate Change. *Scientific Reports* 11, 1 (Nov. 2021), 22320. https://doi.org/10.1038/s41598-021-01714-4
[7] Kevin Cruse, Amalie Trewartha, Sanghoon Lee, Zheren Wang, Haoyan Huo, Tanjin He, Olga Kononova, Anubhav Jain, and Gerbrand Ceder. 2022. Text-mined Dataset of Gold Nanoparticle Synthesis Procedures, Morphologies, and Size Entities. *Scientific Data* 9, 1 (2022), 234.
[8] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. (2024). arXiv:2401.08281 [cs.LG]
[9] Luyu Gao, Xueguang Ma, Jimmy J. Lin, and Jamie Callan. 2022. Tevatron: An Efficient and Flexible Toolkit for Dense Retrieval. *ArXiv* abs/2203.05765 (2022).
[10] Otis Gospodnetic and Erik Hatcher. 2004. *Lucene in Action*. Manning Publications.
[11] Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. 2022. Accelerate: Training and Inference at Scale Made Simple, Efficient and Adaptable. https://github.com/huggingface/accelerate.
[12] Stefan Haefliger, Georg von Krogh, and Sebastian Spaeth. 2007. Code Reuse in Open Source Software. *Management Science* 54, 1 (2007), 180–193.
[13] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient Transfer Learning for NLP. In *International Conference on Machine Learning*. PMLR, 2790–2799.
[14] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale Similarity Search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.
[15] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551* (2017).
[16] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 6769–6781.
[17] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics* 7 (2019), 453–466.
[18] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.
[19] Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A Community Library for Natural Language Processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 175–184. arXiv:2109.02846 [cs.CL]

https://aclanthology.org/2021.emnlp-demo.21
[20] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*. 2356–2362.
[21] Sheng-Chieh Lin, Akari Asai, Minghan Li, Barlas Oguz, Jimmy Lin, Yashar Mehdad, Wen-tau Yih, and Xilun Chen. 2023. How to Train Your DRAGON: Diverse Augmentation Towards Generalizable Dense Retrieval. *arXiv preprint arXiv:2302.07452* (2023).
[22] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods. https://github.com/huggingface/peft.
[23] Mariana Mazzucato. 2021. *Mission Economy.* Penguin Random House.
[24] Thong Nguyen, Sean MacAvaney, and Andrew Yates. 2023. A Unified Framework for Learned Sparse Retrieval. In *Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part III*. Springer, 101–116.
[25] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated Machine Reading Comprehension Dataset. In *CoCo@ NIPs*.
[26] Rodrigo Frassetto Nogueira, Zhiying Jiang, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. *CoRR* abs/2003.06713 (2020). arXiv:2003.06713 https://arxiv.org/abs/2003.06713
[27] Iadh Ounis, Gianni Amati, Vassilis Plachouras, Ben He, Craig Macdonald, and Douglas Johnson. 2005. Terrier Information Retrieval Platform. In *Advances in Information Retrieval*. Springer, 517–519.
[28] David Owen, Dimosthenis Antypas, Athanasios Hassoulas, Antonio F Pardiñas, Luis Espinosa-Anke, Jose Camacho Collados, et al. 2023. Enabling Early Health Care Intervention by Detecting Depression in Users of Web-Based Forums using Language Models: Longitudinal Analysis and Evaluation. *JMIR AI* 2, 1 (2023), e41205.
[29] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Jian Su, Kevin Duh, and Xavier Carreras (Eds.). Association for Computational Linguistics, 2383–2392.
[30] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context Retrieval-augmented Language Models. *arXiv preprint arXiv:2302.00083* (2023).
[31] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. https://arxiv.org/abs/1908.10084
[32] Robby Robson, Elaine Kelsey, Ashok Goel, Sazzad Nasir, Elliot Robson, Myk Garn, Matt Lisle, Jeanne Kitchens, Spencer Rugaber, and Fritz Ray. 2022. Intelligent Links: AI-supported Connections Between Employers and Colleges. *AI Magazine* 43, 1 (2022), 75–82.
[33] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilic, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, Franccois Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-Parameter Open-access Multilingual Language Model. *arXiv preprint arXiv:2211.05100* (2022).
[34] Ben Shneiderman. 2018. *Twin-Win Research.* Springer.
[35] Ben Shneiderman. 2022. *Human-Centered AI.* Oxford University Press.
[36] Yash Raj Shrestha, Georg von Krogh, and Stefan Feuerriegel. 2023. Building Open-source AI. *Nature Computational Science* 3 (2023), 908–911.
[37] Divya Siddarth, Daron Acemoglu, Danielle Allen, Kate Crawford, James Evans, Michael Jordan, and E. Glen Weyl. 2021. *How AI Fails Us.* Technical Report. Justic, Health, and Democracy Impact Initiative & Carr Cenetr for Human Rights Policy.
[38] Jamie Susskind. 2022. *The Digital Republic: On Freedom and Democracy in the 21st Century.* Bloomsbury.
[39] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. *arXiv preprint arXiv:2104.08663* (2021).
[40] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971* (2023).
[41] Paul Trust, Ahmed Zahran, and Rosane Minghim. 2023. Understanding the Influence of News on Society Decision Making: Application to Economic Policy Uncertainty. *Neural Computing and Applications* 35, 20 (2023), 14929–14945.
[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. https://doi.org/10.48550/ARXIV.1706.03762
[43] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Qun Liu and David Schlangen (Eds.). Association for Computational Linguistics, Online, 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6
[44] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *arXiv preprint arXiv:2007.00808* (2020).
[45] Liu Yang, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. 2020. Beyond 512 Tokens: Siamese Multi-depth Transformer-based Hierarchical Encoder for Document Matching. *CoRR* abs/2004.12297 (2020). arXiv:2004.12297 https://arxiv.org/abs/2004.12297