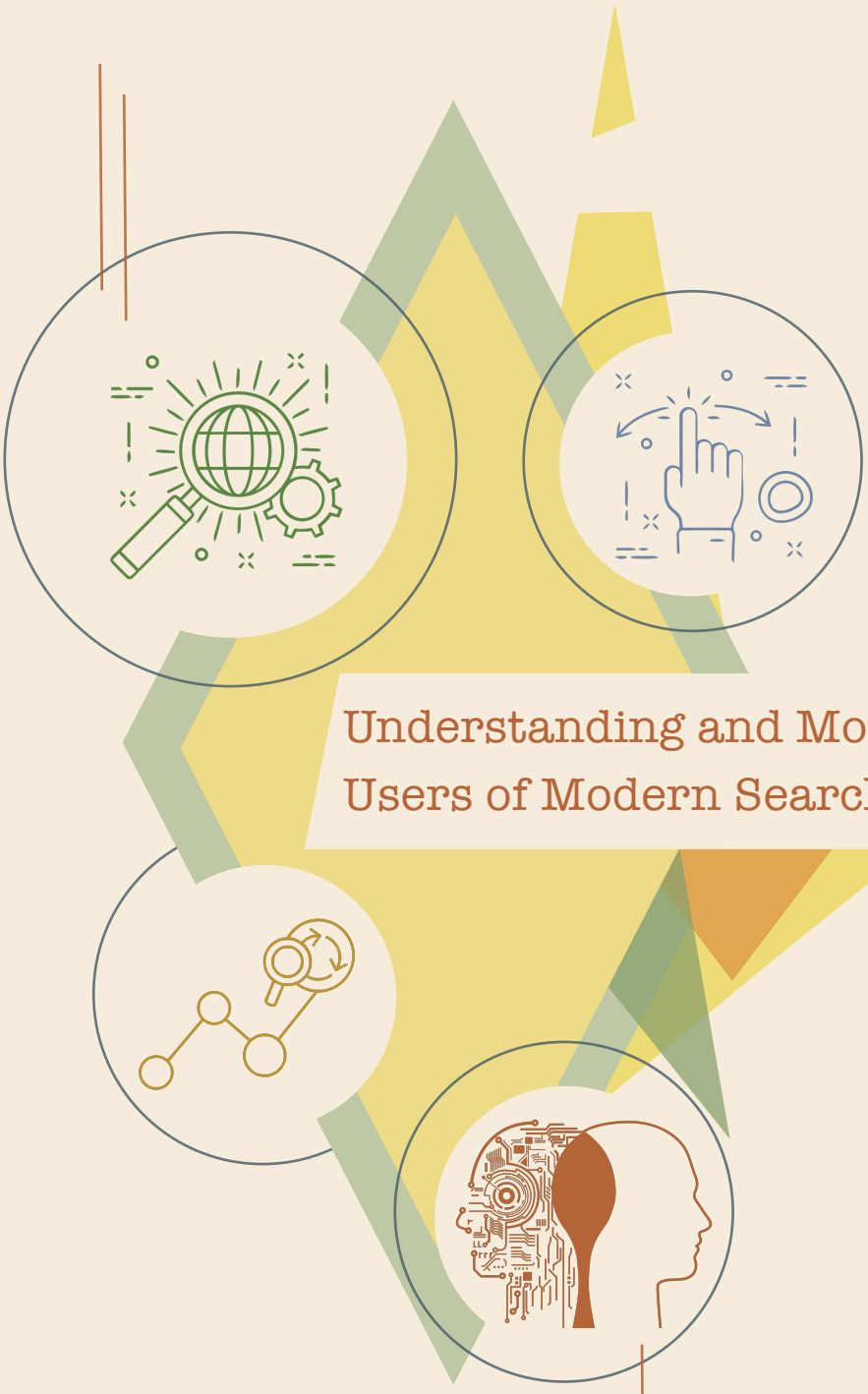


Understanding and Modeling Users of Modern Search Engines

Aleksandr Chuklin

Understanding and Modeling Users of Modern Search Engines

Aleksandr Chuklin

Understanding and Modeling Users of Modern Search Engines

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 Aula der Universiteit
op woensdag 10 mei 2017, te 13:00 uur

door

Aleksandr Chuklin

geboren te Sevastopol, USSR

Promotiecommissie

Promotor:		
	Prof. dr. M. de Rijke	Universiteit van Amsterdam
Copromotor:		
	Dr. E. Kanoulas	Universiteit van Amsterdam
Overige leden:		
	Prof. dr. N. Belkin	Rutgers University
	Prof. dr. T. Joachims	Cornell University
	Dr. ir. J. Kamps	Universiteit van Amsterdam
	Dr. C. Monz	Universiteit van Amsterdam
	Prof. dr. M. Worring	Universiteit van Amsterdam
Faculteit:		
	Faculteit der Natuurwetenschappen, Wiskunde en Informatica	

SIKS Dissertation Series No. 2017-16
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Cover by Samira Abnar and Mostafa Dehghani
Printed by Off Page, Amsterdam
Summary translated by Tom Kenter

ISBN: 978-94-6182-782-1

ACKNOWLEDGEMENTS

I really dislike seeing something 50% done. When I am doing, say, twenty push-ups, I do not praise myself when I have done ten. "Yes," I tell myself, "I am half-way there, but the second half is going to be harder." The same applies to other endeavors, and working on a thesis is no exception. The second half is always harder.

Now that the dissertation is almost complete, I can look back and reflect on its destiny. Life is a peculiar thing: in the middle of the PhD, it brought me to a different country and a different work environment. My thesis work was substantial, but not that close to the finish. I knew it was in this half-done state. To be fair, I was not even sure back then that I wanted to write the thesis text and go through all the formalities of becoming a "Doctor of Philosophy" despite the title sounding nice. I did enjoy staying involved in academic life, though, and this led to more publications. Eventually, I realized that putting a closure to a certain bulk of work would give a sense of accomplishment. As it later turned out, organizing things together in a thesis text was also a useful exercise by itself. Now that I am close to finishing my PhD, I would like to thank plenty of people who were with me during this journey.

First and foremost, I would like to thank my supervisor, Maarten de Rijke. From day one he has always devoted an astonishing amount of time to me. He is so actively involved in all our discussions and paper writing, that it seems that he has an infinite source of energy inside. He always asks the right questions or simply lets me talk and figure out things for myself. Even when I have weeks full of lowlights and very few things to report, I always feel energized and motivated after our meetings. I will always be grateful for having such a great advisor. Thank you, Maarten!

Then, I would like to thank Pavel Serdyukov who was the head of the applied research team at Yandex and de facto my co-supervisor during the first half of my PhD. Pavel has always been good at asking uncomfortable questions. When we shared an office, it was always easy to approach him to discuss ongoing problems and ask for advice. His daily supervision and the unique setup of the research team at Yandex at that time was the reason for great progress I made at the beginning of my PhD. If I think beyond my own work, I would also like to thank Pavel for setting up efficient collaborations with UvA and other universities, and a superb research environment at Yandex. Thank you, Pavel, it was a great pleasure working with you!

I had many great colleagues at Yandex. In particular, I would like to thank those who had the most impact on my thesis and research career. Andrei, my boss at the re-ranking team, has always supplied me with interesting research problems in machine learning and information retrieval. Now that I think about it, he was actually the person who introduced me to the area that eventually became the focus of my thesis. Thank you, Andrei, for all I have learned from you and for being supportive of my move to Pavel's team. I would also like to thank Eugene and Yury from the research team, Mikhail from the snippets team and Lidia from the evaluation team for insightful conversations on the topic of search evaluation and many others. Thank you, guys!

I really enjoy spending time at ILPS each time I go there. It is an amazing place full of smart and funny people. Thank you Abdo, Alexey, Anne, Artem, Caroline, Christophe, Daan, Damien, David, David, Edgar, Evgeny, Fei, Floor, Fons, Harrie, Hendrike, Hosein, Ilya, Isaac, Ivan, Julia, Katja, Katya, Ke, Lars, Manos, Marc, Marlies, Marzieh, Masrour,

Nikos, Petra, Richard, Ridho, Shangsong, Simon, Tatiana, Tom, Wouter, Xinyi, and Zhaochun. Thanks to all of you for interesting discussions and socializing outside of work! And a special thanks to Caroline and Petra for taking care of the admin things.

I would also like to thank my co-authors from ILPS and from all over the world: Adam, Alisa, Anne, Artem, Finde, Floor, Ilya, Katja, Luka, Yiqun, and, of course, Pavel and Maarten. Thank you, for all that you taught me!

It is a great honor for me to have Christof Monz, Jaap Kamps, Marcel Worring, Nicholas Belkin, and Thorsten Joachims on my PhD committee and Evangelos Kanoulas as my copromotor.

Next, I would like to thank my current colleagues: Igor and Süleyman, for helping me to aspire for higher goals, and Enrique and the team for striving to maintain a good balance between research and its real-world applications. In addition, I would like to thank Andrei, Corinna, Dmitry, and Vidhya for providing feedback on my publications.

I also want to thank my teachers from the high-school time. Спасибо Вам, Виктор Фёдорович, за всё, чему Вы меня научили в математике и математическом подходе к решению задач, за подход к изучению языков и просто за разговоры о жизни. Я также очень благодарен Виктору Альбертовичу, Ивану Ивановичу, Людмиле Сергеевне, Марине Викторовне, Роману Анатольевичу и Татьяне Васильевне за всё чему вы меня научили в математике, информатике и физике, а также за вашу поддержку моего участия в олимпиадах. Всё это сильно помогло в моей научной карьере.

Next, I would like to thank excellent professors and TAs from my Bachelor and Master studies. In particular, I would like to thank Andrei M. Raigorodskii for creating a great research group of which I was part during my Bachelor's, and Andrei Romashchenko for being my BSc thesis advisor. I learned a lot from you. In addition, I would like to thank Lena Bunina for all her help during the time of my MSc studies and, of course, for the School of Data Analysis—a unique place where I learned many useful things that were not offered as part of classical university education at the time.

I also want to thank my university friends, in particular, Artem, Egor, Gosha, Gika, Kirill, Max, Nik, and Sasha for all that I learned from you and with you.

Talking about to the thesis text, I would like to thank Mostafa and Samira for designing the cover, and Arina for suggesting interesting design ideas. And a special thanks goes to Tom for helping me with the translation of the summary—to Dutch and to ~~more~~ better English.

Of course, I would like to thank my paranymphs, Alexey and Artem, for helping with numerous small errands and agreeing to be by my side during the thesis defense.

Last but not least, I want to thank my family. В первую очередь, хочу поблагодарить маму за её нескончаемую поддержку все эти годы. Спасибо тебе, мамочка, за всё, чему ты меня научила, за воспитание и мотивацию! Также хочу поблагодарить Ленину семью, в особенности маму, папу, Ксюшу и Таню. Наконец, хочу сказать огромное спасибо моей супруге Лене, за её поддержку и помощь. Невозможно перечислить все твои заслуги, но твоя моральная поддержка и твоя помощь в подготовке презентаций, организации повседневной жизни и поддержании здорового баланса между работой и отдыхом поистине неоценимы!

Contents

1 Introduction

Search is ubiquitous these days. It comes in different forms: from general-purpose commercial search engines used by billions of people, to smaller-scale search interfaces, such as library search or academic search. Driven by both market competition and a desire to make people more productive, modern search engines introduce more and more features to their interfaces: they embed image or video results, augment the search result list by an encyclopedia panel or even make results interactive. Figure 1.1 is an example of such an interface. More and more features are being added, making the *search engine result page* (SERP)[1] increasingly complex. With this complexity come obvious problems of interpreting user behavior and detecting user satisfaction. Was this particular change in the news ranking good for the user? Why did the users click this result so often despite it being objectively bad? What is the best position to put sports results? In order to answer questions like these and make product decisions, we need a solid theoretical ground and models proven to work in practice.

The area of *information retrieval* (IR) was established long before the search engines as we know them today. In fact, the term first appeared in 1950 when Mooers [116] defined information retrieval as the discovery process in a large collection of recorded information. Since then IR emerged as an independent computer science discipline with a number of theoretical and practical pillars. One of the pillars is the so-called Cranfield evaluation paradigm [57]. It established the following evaluation approach. A collection of queries and documents is created where query-document pairs are rated by professional annotators according to the document's relevance to the query. Having such a rated collection, one can evaluate the performance of many different IR systems, where an IR system is a search engine that returns a ranked list of documents as a result of a user query. This idea is a cornerstone of the *Text REtrieval Conference* (TREC).[2] A competition is run during this conference where participants from different research groups compete in how well their IR systems rank unknown documents in response to some unseen queries.

The idea of collecting relevance ratings once and then using them to evaluate many systems is still widely used today. But the IR systems of today are quite different from how they looked fifty or even ten years ago. They no longer return a flat list of homogeneous results: the list is not flat and the results are not homogeneous. Moreover, we can no longer say that the system must put the most relevant documents on top: there are multiple

[1] We maintain a list of notations on page 159.

[2] `http://trec.nist.gov`

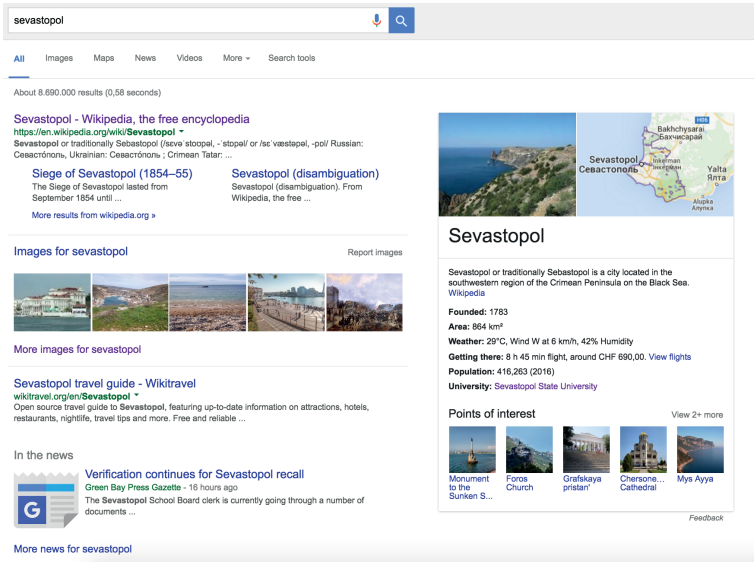


Figure 1.1: An example of a modern complex SERP, featuring Image and News verticals as well as an information panel with encyclopedia data and live flight prices data. Image credits: Aleksandr Chuklin, Google, Wikimedia Commons and its contributors.

aspects of relevance and the notion of "top" is blurred. While not completely discarding the ideas of the Cranfield approach, we suggest taking a step back and *analyze* what defines user behavior and user satisfaction and how one can use this knowledge to *model* users and to make *evaluation* more accurate.

After a search system has been put online, one of the easiest type of user interaction data to collect is clicks. Clicks are often interpreted as a signal of user satisfaction. However, there are whole classes of situations where we observe no clicks at all, yet the users are satisfied. These are called *good abandonments* [37, 104, 161]. We need to understand when such situations occur, how users behave and why they behave in a certain way. To achieve that, we must go beyond clicks.

In general, *modeling* the users of a search system is an important task with many applications. The first application is understanding the users through fitting a model to the user data [69]. A second application is using a user model to perform simulated experiments when no or limited user data is available [48, 82]. Thirdly, user modeling is used to improve ranking [25] and, finally, to build more accurate evaluation metrics [42, 112]. Contemporary SERPs pose two big challenges to user modeling: taking into consideration various signals beyond clicks (such as mousing or scrolling) and accounting for the heterogeneous nature of results and their non-trivial arrangement on a SERP.

While user interaction data helps us to build more accurate Cranfield-style evaluation metrics through user modeling, it may also be used as a form of direct, albeit implicit, relevance feedback. Out of two popular methods, A/B-testing [99] and interleaving [27], the second was shown to have higher sensitivity. Interleaving is an online evaluation strategy that compares two ranking systems—usually the current production ranking

and an experimental ranking—by mixing results from the two systems into one and interpreting users' clicks as implicit feedback. While such a method has high sensitivity and intuitive interpretation of the click scores, it nevertheless faces challenges when heterogeneous SERPs are involved as it is not a straightforward task to interleave two SERPs featuring different special elements and potentially having different layouts.

In the next section we formulate the main research questions of the thesis as well as the sub-questions that aim to go deeper on each one of them.

1.1 Research Outline and Questions

Complex SERPs create many challenges in different areas of information retrieval. Below we list the research questions we will be answering in this thesis. We first list the bigger questions and then dig deeper into each individual one.

RQ1 Can we substantially improve existing click models by taking into account result page structure and aggregated user characteristics?

RQ2 How do we evaluate click models?

RQ3 How can we make use of click models to improve offline evaluation metrics?

RQ4 How can we improve user models and offline evaluation metrics to account for non-trivial attention patterns and direct usefulness of result snippets?

RQ5 How can we perform accurate online quality evaluation on complex SERPs without affecting the user experience?

1.1.1 Click Models for Complex SERPs

While actual users should always be the golden standard as a source of behavioral data, sometimes we simply lack user interaction data. Or it can well be the case that the study we want to conduct has risks of degrading the user experience and we just cannot afford such risks and have to forgo experiments with real users. In other words, collecting data from users does not scale in some cases. That is why we need a solid model of the user interacting with the SERP.

For this purpose we employ the notion of a *click model*, a probabilistic model that predicts next user actions given historical data [45]. In previous work only clicks were considered as interaction data. However, recent work by Huang et al. [88] goes beyond that and adds mouse movements as an additional source of information. We want to continue this line of work and make use of other sources of information to improve the model accuracy. As was demonstrated in prior work [51, 154], some results may be more visually salient (e.g., video results) and hence attract more clicks. We want to leverage that observation and put the SERP layout information as well as our knowledge about user intents into a click model.

The main sub-questions we want to answer are:

RQ1.1 How can we use page layout information to improve click models?

RQ1.2 How can we use aggregated user characteristics such as vertical orientation to improve existing click models?

1.1.2 Evaluating Click Models

There are different ways to evaluate click models. Some of them are motivated by the theory behind the *probabilistic graphical model* (PGM) framework that we use, some are motivated by the applications of click models.

Firstly, we want to know how different click models that are widely used these days compare to each other when measured across different evaluation dimensions on a common dataset. Previously the models were mostly evaluated on various proprietary datasets. Moreover, different sets of metrics were used to evaluate different models. Our goal is to establish a common ground.

Secondly, we want to question whether the current set of click model evaluation metrics is sufficient to evaluate click models for modern SERPs. We suggest a complementary evaluation approach that allows to better understand quality of such click models.

We formulate the above as the following research questions:

RQ2.1 How do different click models perform when evaluated on a common dataset?

RQ2.2 How should we evaluate click models for complex aggregated SERPs?

1.1.3 Click Model-based Metrics

Search evaluation is a very important but complex issue. It is always hard to justify a new evaluation metric. That being said, we argue that the need for better evaluation techniques for contemporary SERPs should be put in our research agenda.

After having built efficient models of user click behavior, we suggest using these models to build Cranfield-style evaluation metrics that are more aligned with the actual users. In particular, we answer the following questions:

RQ3.1 Can we make use of click models to build better evaluation metrics? How do such click model-based IR metrics differ from traditional offline metrics?

RQ3.2 Which evaluation metrics are better tied to the user? Do click model-based metrics show higher agreement with online experiments? How do they compare in terms of discriminative power?

RQ3.3 How well do different offline metrics perform in the presence of unjudged documents?

RQ3.4 How can we modify offline metrics to enhance agreement with online experiments?

RQ3.1 requires developing a framework to derive an evaluation metric from a click model. It is also important to show that the resulting metrics differ from already existing and widely used metrics such as *discounted cumulative gain* (DCG) [89] or *expected reciprocal rank* (ERR) [26], which is what the second part of the question is about.

After suggesting new evaluation metrics with a solid foundation of a click model, we want to show that they not only differ from existing metrics, but also outperform them. So in **RQ3.2** we adopt a commonly used discriminative power analysis to compare metrics. On top of that we suggest measuring a metric's performance by looking at how well it agrees with the outcome of the online click-based evaluation. Since online experiments are often used as a rather time-consuming but precise tool to make final decisions, it is important that during many iterations of algorithm development, one has a metric that would be better aligned with the final outcome of the online experiment.

RQ3.3 touches on an important question of missing relevance judgements. As was previously pointed out by Sakai [131], we often face a situation where part of the documents that an IR system ranks at the top of a SERP lacks relevance judgements, in other words, it was not judged by the raters. Due to a limited amount of resources, this situation happens rather often and we need to check how our metrics perform in such a scenario.

Finally, **RQ3.4** suggests ways to improve the quality of evaluation metrics (measured by the agreement with the outcomes of the online experiments), especially in case judgements may be missing.

1.1.4 Going Beyond Clicks

As result pages become more and more complex, two big challenges for evaluation arise: non-trivial attention patterns and good abandonments. As was shown by Clarke et al. [51], Sushmita et al. [154], some results are more visually salient, e.g., video or image results, and therefore attract more clicks (and, in general, more of the user's attention). There also exists the opposite effect—a SERP may already contain an answer to the query and does not require clicks. This effect is known as *good abandonment* [30, 104].

We argue that the notion of good abandonment also applies to individual SERP items, not just to the SERP as a whole. By analyzing factoid queries we show that the quality of individual result snippets has a peculiar connection to the abandonment rate, suggesting there may be direct utility gain from examining results on a SERP.

Combining the ideas of non-trivial attention pattern and direct usefulness of result snippets, we suggest a novel evaluation metric that more accurately predicts user satisfaction. Since our metric has a user model behind it, we ask separate questions about the model and the metric based on it:

RQ4.1 Does a model that unites attention and click signals give more precise estimations of user behavior on a SERP and self-reported satisfaction? How well does the model predict click vs. satisfaction events?

RQ4.2 Does an offline evaluation metric based on such a model show higher agreement with user-reported satisfaction than conventional metrics such as DCG?

1.1.5 Online Evaluation

In the area of online evaluation, we concentrate our effort on a family of methods called *interleaving*. As was shown by Hofmann et al. [84], Radlinski and Craswell [123], evaluation methods based on relative techniques such as interleaving are easier to interpret and are more sensitive than evaluation methods based on comparing absolute click metrics.

In the case of modern complex SERP we aim to achieve the efficiency of conventional interleaving methods while preserving the conventional aggregated search user experience. This proves to be a challenging task due to interleaving algorithms' nature to "mix up" results. Consequently, we want to formulate our research question as follows:

RQ5.1 **Influence on the user experience.** What effect do different interleaving methods—both conventional and newly introduced vertical-aware—have on the user experience in the case of complex SERPs? Do any of these methods run the risk of degrading the quality of the results or altering the user experience?

RQ5.2 **Correctness and sensitivity.** Do different interleaving methods always draw correct conclusions about the better system? How fast, in terms of the number of impressions and amount of feedback needed, can they detect that one aggregated search system is to be preferred over another?

RQ5.3 **Unbiasedness.** Do the interleaving methods that we consider provide a fair and unbiased comparison, or do some of them erroneously infer a preference for one aggregated search system over another in situations where implicit feedback is provided by a randomly clicking user?

In order to answer **RQ5.1** we consider different pairs of complex SERPs and analyse the effect on the user experience imposed by the interleaving methods that we consider. **RQ5.2** concerns the ability of an interleaving method to correctly capture the difference between two rankers by using the minimal amount of implicit user feedback. Finally, to answer **RQ5.3** we need to check that none of the evaluation methods infer statistically significant preferences when no preference is expected.

1.2 Main Contributions

Here we summarize the main contributions of the thesis, roughly split into three groups: theoretical, algorithmic and software contributions.

1.2.1 Theoretical Contributions

Our main theoretical contributions lie in the area of evaluating evaluation methods and user models. Apart from that, we introduce the idea of per-intent parameters for click models and a Bayesian framework to estimate click probabilities in this case. Below we describe the contributions in more detail.

T1 Bayesian framework to estimate click probabilities using per-intent parameters of a probabilistic graphical model (Chapter 4).

T2 A novel evaluation method for click models—click model intuitiveness (Chapter 5).

T3 An evaluation framework for offline metrics based on their agreement with the outcomes of online experiments (Chapter 6).

T4 An evaluation framework for offline metrics based on their agreement with user-reported satisfaction (Chapter 7).

T5 A new evaluation facet for online interleaving experiments—method's influence on the user experience (Chapter 8).

1.2.2 Algorithmic Contributions

Algorithmic contributions form the core of the thesis. In all but one research chapters we suggest concrete algorithms that can be directly applied to existing search engines to make their operation measurably better.

A1 A method to convert any click model into an intent-aware click model—a model that accounts for complex SERP structure and distribution of user intents (Chapter 4).

A2 A method to extend any click model beyond the first page of results (Chapter 4).

A3 A method to derive evaluation metrics from click models (Chapter 6).

A4 A user model and an evaluation metric that unite clicks, attention and satisfaction (Chapter 7).

A5 A method to extend two popular interleaving methods so that they become suitable for complex SERPs (Chapter 8).

1.2.3 Software Contributions

Open-source software is an important part of our contribution. We contribute in two ways: (1) by releasing new software libraries, and (2) by contributing to existing projects:

S1 We release a library for training and using click models and click model-based metrics: `https://github.com/varepsilon/clickmodels`.

S2 We contribute click model implementations to two other commonly used libraries, namely `PyClick`[3] and `Lerot`.[4]

S3 We release a collection of components used to train and evaluate our CAS model (see Chapter 7): `https://github.com/varepsilon/cas-eval`. It contains the following components:

- a configuration settings for a search proxy;
- client and server code to collect user interaction data via proxy;
- an interface for managing the search log of a user;
- templates and post-processing scripts for relevance crowdsourcing;
- code to train and evaluate CAS model and metric.

[3] A Python library to work with click models, `https://github.com/markovi/PyClick`.
[4] A learning-to-rank library, `https://bitbucket.org/ilps/lerot`.

Our `clickmodels` project S1 is a self-sufficient library we released. It has the new models we develop in Chapter 4, the machinery to train parameters of a click model-based metric (Chapter 6) and a method to use click model parameters as ranking features similar to [25]. Apart from that, it has the basic machinery to work with existing click models as well as sufficient flexibility to build new models on top of it. For example, PSCM model by Wang et al. [160] was built on top of our framework.[5]

The contribution to `PyClick` [45] includes implementations of some click models, e.g., DBN [25] and CCM [74], as well as some changes throughout the framework, ensuring it is currently the most comprehensive, extensible and easy to use library for working with click models.

Our contribution to `Lerot` [137], an online learning to rank framework, includes porting some state-of-the-art click models to be used to simulate users as well as implementing vertical-aware interleaving methods that we introduce in Chapter 8.

1.3 Thesis Overview

Below we present an overview of the thesis structure and give recommendations for reading directions.

Chapter 1 gives an introduction to the thesis. It starts with a general motivation for the problem we study, outlines the research questions in Section 1.1, summarizes the main contributions in Section 1.2, gives overview of the thesis in Section 1.3 (current section) and lists the origins in Section 1.4.

Chapter 2 gives necessary background and discusses related work.

Chapter 3 introduces some common baselines and basic evaluation metrics used in Chapters 4–8.

Chapter 4 talks about click models for web search—probabilistic models of user behavior on a SERP. In that chapter we introduce several new models aimed at modeling user behavior on a modern complex SERP.

Chapter 5 focuses on evaluating click models. In that chapter we introduce a new evaluation method specifically geared towards click models for complex SERPs and present a comprehensive evaluation of basic click models on a number of evaluation criteria.

Chapter 6 introduces a family of offline evaluation metrics based on click models. It also establishes a new way of evaluating evaluation metrics based on their agreement with online experiments.

Chapter 7 goes one step further and enhances evaluation metrics with implicit and implicit signals related to user attention and user satisfaction. In that chapter we also introduce an evaluation approach where metrics are compared to each other in terms of how well they predict user-reported satisfaction.

Chapter 8 approaches the problem of evaluating the user experience from a different angle, namely it discusses the settings where user experience is evaluated online using interleaving. We introduce a framework to evaluate online evaluation methods and suggest two new interleaving methods that can be used in the setting of complex SERPs.

[5] Their code is available at `https://github.com/THUIR/PSCMModel`.

Chapter 9 gives a summary of conclusions for the research chapters and links back to the research questions introduced in Section 1.1. We also discuss limitations and potential future directions.

Readers familiar with the basics of IR system evaluation and click modeling may skip the corresponding parts of the background material offered in Chapters 2 and 3. Chapters 4–6 require basic familiarity with click models, but are independent from each other and can be read in any order. Chapter 7 largely builds on the material introduced in Chapter 6 and should be read after it. Finally, Chapter 8 can be read independently of Chapters 4–7 and only requires familiarity with basic IR evaluation methods introduced in Chapter 3.

1.4 Origins

Below we list publications which form the basis for each research chapter. We use abbreviations for the author names which should be self-explanatory.

Chapter 4 is based on the paper titled *Using Intent Information to Model User Behavior in Diversified Search* by A. Chuklin, P. Serdyukov, and M. de Rijke [41] published at ECIR'13. The algorithm was designed by AC, experiments and analysis were performed by AC. All authors contributed to the text, AC did most of the writing.

It is also based on the article titled *Modeling Clicks Beyond the First Result Page* by A. Chuklin, P. Serdyukov, and M. de Rijke [43] published as a short paper at CIKM'13. The algorithm was designed by AC, experiments and analysis were performed by AC. All authors contributed to the text, AC did most of the writing.

Chapter 5 is based on the paper titled *Evaluating Intuitiveness of Vertical-aware Click Models* by A. Chuklin, K. Zhou, A. Schuth, F. Sietsma, and M. de Rijke [44] published as a short paper at SIGIR'14. All authors contributed to the evaluation idea. AC and AS contributed equally to the model design. KZ did most of the experiments and analysis. All authors contributed to the text, AC did most of the writing.

It also includes some evaluation experiments from the *Click Models for Web Search* book by A. Chuklin, I. Markov, and M. de Rijke [45] published by Morgan & Claypool in 2015. All authors contributed to the synthesis of the material and, respectively, to the text. AC and IM performed the experiments and analyzed the results.

Chapter 6 is based on the paper titled *Click Model-based Information Retrieval Metrics* by A. Chuklin, P. Serdyukov, and M. de Rijke [42] presented at SIGIR'13. The method was designed by AC, experiments and analysis were performed by AC. All authors contributed to the text, AC did most of the writing.

Chapter 7 is based on the paper named *Incorporating Clicks, Attention and Satisfaction into a Search Engine Result Page Evaluation Model* by A. Chuklin and M. de Rijke [35] presented at CIKM'16. The method was designed by AC, experiments and analysis were performed by AC. Both authors contributed to the text, AC did most of the writing.

It also draws motivation from the paper titled *Good Abandonments in Factoid Queries* by A. Chuklin and P. Serdyukov [37] presented as a poster at WWW'12. The method was designed by AC, experiments and analysis were performed by AC. Both authors contributed to the text, AC did most of the writing.

Chapter 8 is based on the journal article, *A Comparative Analysis of Interleaving Methods for Aggregated Search* by A. Chuklin, A. Schuth, K. Zhou, and M. de Rijke [48] published at TOIS in 2015, which itself is an extended version of the paper titled *Evaluating Aggregated Search Using Interleaving* by A. Chuklin, A. Schuth, K. Hofmann, P. Serdyukov, and M. de Rijke [40] published at CIKM'13. AC suggested the interleaving methods, implemented them and did most of the experiments and analysis. AS did most of the simulation-based experiments and their analysis in the original CIKM'13 version of the paper. KH contributed most to the analysis of the unbiasedness results in the original CIKM'13 version. Most of the new experiments for the extended TOIS'15 version, including those based on simulations, were done by AC. All authors contributed to the text, AC did most of the writing.

Apart from the papers listed above there were several other publications that contributed to this thesis. Multiple tutorials given at SIGIR'15, WSDM'16 and RuSSIR'16 [46, 47, 49, 50] as well as the co-authored paper [73] presented at CLEF'15 helped to shape the story of click models and their applications. Work on good abandonments [38, 39] helped to understand the problem better and gain motivation for need to go beyond clicks in Chapter 7, while [33] determined the design of the metric used in that chapter.

Work on other aspects of aggregated search [34, 113] helped in understanding the problem of modern complex SERPs better, while work on result filtering [36] was a useful diversion to a different area of information retrieval. Finally, work on file synchronisation [32] from the area of information theory helped in forming general computer-science thinking and research-related programming skills early on.

2 Background

In this chapter we discuss the required background and the previous work that the thesis builds upon. We start with a general introduction to information retrieval in Section 2.1, then move to areas of particular importance for the thesis: vertical search in Section 2.2, user modeling in Section 2.3, offline evaluation in Section 2.4 and online evaluation in Section 2.5.

We then study the main basic concepts in greater detail in Chapter 3.

2.1 Information Retrieval

The term *information retrieval* (IR) was introduced by Mooers [116]. He advocated for it to become a separate area of research, distinct from database research. And he was right, an ACM Special Interest Group on Information Retrieval was formed and now organizes a yearly SIGIR conference along with many other conferences and forums; there are also other groups around the globe supporting research in IR.

The initial focus for IR was on indexing and matching [162]. Similar to a book that has an index of terms in the back, one can build an *index*[1] of the words appearing in a collection of documents. If a user comes with an information need expressed by a query, we can split the query by words (query terms) and retrieve the documents from the index where each of the query terms appear. Many optimizations have been proposed for matching and retrieval since then [176] and there is still work being published on optimizing this first step of the search process (see, e.g., [17]).

With the rapid growth of the size of document collections, especially the World Wide Web, *ranking* retrieved documents became an important problem. It was often no longer possible for a user submitting a query to examine all documents returned. Even when it is possible, it is a tedious job that can be helped by algorithms. Apart from that, the users are often unable to precisely specify their information need as a bag-of-words query [14], so a search engine should aid them in the process of information discovery. To address these issues search engines—or *IR systems* as we often call them—started ranking documents according to their match to the user's query. One of the simplest approaches to ranking the retrieved documents is to order them by the *term frequency* (TF), the number of times the query terms appear in the document [110]. Another ranking signal is the *inverse document*

[1] Sometimes referred to as *inverted index*.

frequency (IDF), one over the number of documents the term appears in. Intuitively, if the term appears only in a handful of documents, it is likely to be an important term for that query. Conversely, if a term appears in many documents it is likely to be a common word, such as an article or the verb "to be." To use this signal for ranking, one multiplies the TF score by it to get the so-called TF.IDF score [150].[2] Both TF and TF.IDF scores are still used these days, albeit as part of a bigger ranking model.

With the development of more complex ranking functions such as *best match 25* (BM25) by Robertson et al. [128] it became necessary to evaluate the quality of IR systems. The first systematic study of rating query-document pairs and comparing ranking functions was done in the context of the Cranfield project [55–57] which we cover in more detail in Section 2.4.

The next step after ranking results is their presentation on a *search engine result page* (SERP). Instead of just showing a list of links, nowadays we augment them with document titles and small snippets extracted from the document and biased towards the query [136, 156]. As was shown by Dumais et al. [68], it is also helpful to organize results on the page by their sub-topics because it facilitates the information consumption. This idea was adopted in what is now called *aggregated* or *vertical* search and is closely related to the notion of federated search. We discuss vertical search in more detail in Section 2.2.

After results are presented to the user and the user interacted with them, their actions are stored in a search log and can be analyzed. Apart from observational studies, the search log can be used to build a model of user behavior (Section 2.3) or to interpret user actions as implicit quality signals, thus performing so-called online quality evaluation (Section 2.5). More on that in the corresponding sections below.

For a more comprehensive overview of the history of IR one can refer to [134]. A general introduction to the area is well described in a book by Manning et al. [111] as well as some earlier work by Spärck Jones [149] and Baeza-Yates et al. [13].

2.2 Vertical Search

The first topic we will dive deeper into is the so-called *vertical search*, a paradigm where the SERP is composed of results coming from different verticals, such as an Image, News or Web vertical. Vertical search is also sometimes referred to as *aggregated search* (AS) or even *federated search*, although one may argue that the latter is more concerned with selecting and routing queries to vertical search engines rather than aggregating results on a SERP (see, e.g., [60, 120, 144]). We claim that the paradigm of a heterogeneous SERP aggregated from different verticals has become de-facto standard of modern web search and therefore the topics we discuss in this section resonate well with the rest of the material throughout the thesis.

Dumais et al. [68] demonstrated that users are more efficient with their goals whenever search results are organized according to sub-topics. Moreover, Radlinski and Dumais [125] showed that it is useful to diversify results returned to the user. A good example is a query `[jaguar]` which may refer to an animal, car brand or, at some point in history, to

[2] In practice, IDF undergoes log-transformation to avoid skewness.

an operating system. Showing results for all those distinct *intents* helps to improve overall user satisfaction.

Classical *vertical search* deals with organizing results coming from different specialized ranking systems (*verticals*) into one unified SERP. If the results from a vertical are deemed to be suitable for the query, they are placed in a grouped manner somewhere on a SERP. Usually there are three or four insert positions—slots—where vertical results can be placed [121], although some search engines do not follow this practice [152].

The vertical search problem is usually split in two tasks: vertical selection and vertical ranking. *Vertical selection* deals with determining which verticals are relevant to the given query, while *vertical ranking* is the problem of deciding which vertical block should be placed higher than others. There are several papers following the vertical approach that aim at solving these tasks (e.g., [5–7]). The problem of aggregated search evaluation also received some attention [8, 122, 171].

An alternative approach where vertical results are all ranked together with an objective of optimizing diversity in addition to relevance, also attracted some attention in both academia and industry [1, 153].

The search engine that introduced such aggregated (or faceted) search interface early on was Naver.[3] There, a SERP can contain more than ten results per query—in fact, many more—and results are always organized into separate groups, one for each vertical [142]. Yahoo! Search[4] and Bing[5] historically inserted blocks of vertical documents on fixed positions (slots) in addition to the organic web search documents [7, 122]. Yandex[6] allows vertical blocks to appear in any position [41], and the same appears to hold for Google.[7]

2.3 User Modeling

As we have mentioned already, by processing a search log, we can come up with a model of user behavior that can be used to simulate user's actions. Clicks are a type of interaction that appear on both desktop and mobile search and are easily accessible for any search engine. Therefore, we are going to focus first and foremost on *click models*, although there are now some early attempts to predict user mousing [62] instead of clicks which we also employ in our work (see Chapter 7).

Perhaps the first work that used the term "click model" was the paper by Craswell et al. [59] in which the *cascade model* (CM) was introduced. This work also considered a number of other models that are based on the *position bias* hypothesis—an observation that results lower in the rank get less attention and, consequently, fewer clicks. The position bias had previously been established by Joachims et al. [92] using eye tracking and click log analysis. Prior to the work of Craswell et al. a couple of models that we now

[3] `http://www.naver.com`, the most used search engine in South Korea as of 2015.

[4] `https://search.yahoo.com`, a global search engine with a significant market share in Hong Kong and Taiwan as of 2015.

[5] `https://www.bing.com`, a global search engine with a significant market share in the United States as of 2015.

[6] `https://yandex.com`, the most used search engine in Russia with a significant market share in several other countries as of 2015.

[7] `https://www.google.com`, a global search engine with more than a billion monthly active users as of 2015.

call click models had been introduced for the sake of ranking evaluation [70, 115].

Following the publication of [59], the *user browsing model* (UBM) was introduced by Dupret and Piwowarski [69], the *dynamic Bayesian network* (DBN) model by Chapelle et al. [26], and the *dependent click model* (DCM) by Guo et al. [74]. Most click models that have been introduced since then are based on one of those three models. We discuss these and several other models in more detail in Section 3.1.

For parameter estimation, mostly *maximum likelihood estimation* (MLE) and *expectation maximization* (EM) are used (see, e.g., [45]), but there is some work that explores other methods. For example, Liu et al. [106] used the same assumption as in the UBM model, but proposed a faster inference algorithm. Zhang et al. [168] suggested a so-called *deep probit Bayesian inference* for more accurate estimation that uses query-document matching scores, while Shen et al. [143] suggested to leverage existing work on matrix factorization and collaborative filtering for more accurate estimation of model parameters for previously unseen query-document pairs.

In more recent years, quite a few improvements to the basic click models have been suggested. For instance, once search engines began to present their search results as blocks of verticals aggregated from different sources (see Section 2.2), click models for aggregated search started to receive their rightful attention [28, 159]. We also contribute to this effort in Chapter 4.

Overall, the development of click models goes hand in hand with their applications. For example, recent research on analyzing mouse movements has been applied to improve click models in [88]. And as web search is rapidly becoming more personalized [66, 155], this is also reflected in click models; see, e.g., [163]. Another direction is taken by Zhang et al. [169], who go beyond a single query by modeling user tasks in a so-called *task-centric click model*. We believe these directions are going to be developed further, following the evolution of search engines.

Above we briefly discussed background material on user models, mainly click models. User models are the main object of study in Chapters 4 and 5, while Chapters 6 and 7 focus on a particular application thereof, namely user model-based offline evaluation models. Finally, we also use click models in Chapter 8 to simulate users. For a more technical overview of the basic click models we refer to Section 3.1. For a comprehensive survey on click models one can also consult [45].

2.4 Offline Evaluation

Cranfield-style evaluation, as put forward by Cleverdon et al. [55–57], assumes the following protocol to evaluate ranking quality of a search engine. First, collections of queries and documents are created. Then documents are rated by independent judges—*assessors*—to determine how well the document matches the query. In the simplest case the ratings are 0 or 1, that is, the document is either irrelevant or relevant to the query. Nowadays graded relevance scores are often used, usually three or four grades, e.g., *Irrelevant*, *Marginally Relevant*, *Relevant*, *Perfect Match* are the names we use for the relevance grades in Chapter 7. For each new ranking function we then compute some quality metric based on the relevance scores assigned by assessors and average it across

the query collection. An example of such a quality metric would be the average number of relevant documents amongst the top ten returned by the system (see Section 3.2.1).

The benefit of this approach is that we can try many new ranking functions without the need to repeat the time-consuming and costly process of collecting ratings. This allows us to have many iterations of ranking function development without the need to try each iteration on real users. We can even optimize our ranking function directly to maximize a metric. This is why we call such metrics *offline metrics*, that is, they do not require us to deploy the system live, as opposed to online metrics discussed in the next section.

Assessors can be trained judges doing this work for a long time. Alternatively, assessors can be crowd workers, sourced by one of the crowdsourcing platforms [4], such as Amazon Mechanical Turk,[8] CrowdFlower[9] or Yandex.Toloka.[10]

As document collections become larger, the task of collecting ratings for all query-document pairs becomes infeasible. In that case a technique called *pooling* is applied [148], whereby top-N documents retrieved by several ranking systems are sent for rating. To avoid being stuck with the same documents, this technique is best applied in a living evaluation lab setting, where new systems are constantly added, the pool is updated and the new documents are rated. There are multiple avenues for such data collection, most popular are the benchmarking efforts run as part of various conferences: TREC[11] in North America, CLEF[12] in Europe and NTCIR[13] in Asia. The competitions run regularly, with teams from various institutions submitting the output of their ranking systems and thus providing candidates for pooled rating collection.

Even when the new judgements are constantly collected, we often have to deal with missing relevance judgements [18, 20, 131] and the fact that some documents may never be seen by raters. This suggests that alternatives to pooling should be sought, and, in fact, there was some work exploring this direction [19, 23, 105, 135]. We also consider the problem of missing judgements when analysing metrics in Chapter 6.

The creation of traditional test collections against reduced costs has received considerable attention. Carterette and Allan [22] and Sanderson and Joho [135] discussed approaches to building test sets for evaluation at low cost. Azzopardi et al. [12], Berendsen et al. [15] went a step further and described methods for automatically generating test collections and training material for learning-based rankers, respectively. Buckley and Voorhees [18] introduced a new metric called *bpref* to use in a setup where we have missing relevance judgements. Sakai [131] proposed an alternative solution that does not require a new metric. Apart from evaluation metrics, there are other interesting problems arising when dealing with large query sets; these are addressed by the TREC Million Query Track [3] and further studied by Carterette et al. [24].

On the metric side, precision and recall have long been standard metrics in computer science and statistics. Then, metrics designed for ranked lists such as average precision and *discounted cumulative gain* (DCG) started to emerge, which we cover in more detail

[8] `https://www.mturk.com`, as of 2016 limited to researchers from Australia, Canada, the United Kingdom and the United States.

[9] `https://www.crowdflower.com`, available to researchers from most of the world as of 2016.

[10] `https://toloka.yandex.com`, available to researchers from most of the world as of 2016.

[11] Text REtrieval Conference.

[12] Conference and Labs of the Evaluation Forum.

[13] NII (National Institute of Informatics) test collection for information resources.

in Section 3.2. More recently metrics for diversification and aggregated search have been suggested. If we assume that each document has separate relevance labels for different query intents, we get a family of intent-aware metrics by Agrawal et al. [1] as well as other metrics addressing the problem of diversity: ERR-IA by Chapelle et al. [26], α-NDCG by Clarke et al. [52], and D♯-NDCG by Sakai and Song [133]. ERR-IA has become a standard metric for the TREC diversity track [53]. These diversity-aware metrics can also be adopted to the aggregated search settings. Alternatively, one can use a dedicated aggregated search metric such as the *aggregated search metric based on rank-biased precision* (AS_{RBP}) [171].

Above we discussed the history and the main areas of research for offline search evaluation. We discuss several widely-used metrics in Section 3.2. For a survey-like overview one may refer to [94].

2.5 Online Evaluation

While offline evaluation has its advantages and is still widely used, there exist another method, namely running experiments with real users. Sometimes it is the only way of doing studies due to the sensitivity of the data (e.g., one cannot use assessors to rate results of an email search engine due to obvious privacy concerns) or simply because rating documents would be too expensive or too slow for a small search engine to afford. And, perhaps, the most important advantage of online evaluation is that one can often interpret the signals unambiguously where the users serve as final judges of quality—if the users engage with the search more often, or, perhaps, buy more (in the case of a product search engine), it is a clear signal of search quality improvement.

There are two lines of work addressing the problem of online search quality evaluation. First is a pairwise *A/B-testing* approach (see, e.g., [99]), where two different versions of the system (called A and B) are deployed to a small percentage of the user base, say, 1% each, and the user behavior is observed. We should stress that this type of experiment is used widely and not just for search engine evaluation: one can use it to compare two versions of virtually any system, also outside information and communication technology. The trick is how to interpret user behavior and decide which system should win the comparison. For search engines looking just at total number of clicks—*click-through rate*s (CTRs)—would be wrong, because users are subject to different kinds of bias as was shown by Joachims et al. [93]. Abandonment rate as well as other simple SERP-level signals (see Section 3.3) are widely used, although, they are not able to capture long-term user retention. On the other hand, experiments with user retention as a metric take long to converge as its sensitivity is relatively low [100, 101]. Recently several methods have been suggested that are aimed at improving sensitivity of click metrics and at optimizing the schedule of A/B-testing experiments [61, 67, 97, 98].

Another line of work is *interleaving*. Unlike A/B-testing where two different versions of the SERP are shown to different users, in interleaving experiments we mix the results of the two systems in the same SERP and show it to a single user. This setup helps to greatly improve sensitivity over absolute online metrics because we eliminate variance due to user differences [27, 84]. Before the word "interleaving" was coined by Radlinski et al. [127], the method itself was described in [91]. Several other algorithms aimed at interpreting

user clickthrough data for evaluation and learning were suggested [90, 126], but these alternative evaluation approaches did not gain as much popularity as interleaving.

The two most commonly used interleaving methods are *team-draft interleaving* (TDI) [127] and *balanced interleaving* (BI) [91]. TDI can be described as follows. For each user query we build an interleaved list L whose documents are contributed by rankings A and B, the two rankings that we want to compare. This interleaved list is then shown to the user and the user's clicks are recorded. The system that contributes most of the documents clicked by the user is inferred to be the winner for the particular query-user pair; the system that wins for most such pairs is then considered to be the better system. BI uses a different algorithm to build an interleaved list L and a more sophisticated procedure for determining the winner. These interleaving methods, as well as their modifications, were extensively studied in [27, 83].

Other interleaving methods include *document constraints interleaving* (DCI) by He et al. [79] and *probabilistic interleaving* (PI) by Hofmann et al. [80]. DCI produces interleaved lists similar to BI, but it has a different and more involved way of determining the winner by computing which ranking violates the smallest number of constraints. PI has the advantage that historical interaction data can be easily reused using importance sampling, for instance in an online learning to rank setting [82] or potentially also in an evaluation setting. However, because PI relies on probabilistic rankers, it risks showing the user results of the poorly performing rankers that are not related to the original rankers being interleaved, which may affect the user experience [83].

A more recent approach to generating interleaved pages, called *optimized interleaving* (OI), was proposed by Radlinski and Craswell [124]. This method does not have the drawbacks of PI. Unlike TDI, it first enumerates all the possible interleaved lists and then assigns probabilities, according to which an interleaved list is drawn. This probability distribution is selected such that the comparison is unbiased and such that it has optimal sensitivity.

Finally, Schuth et al. [138, 139] suggested extending interleaving beyond two systems obtaining a method called *multileaving* where multiple systems are compared at once, therefore allowing for faster iterations of quality improvements and faster online learning [141].

Above we discussed different approaches to evaluating system's quality online. Our Chapter 8 is entirely devoted to online evaluation and its adaptation to the case of complex heterogeneous SERPs. In Chapter 6 we compare offline metrics to online quality signals, while in Chapter 7 we use online metrics to demonstrate that looking only at clicks is insufficient to understand user's satisfaction. Below, in Section 3.3 we detail on the absolute online evaluation metrics that we use later in the thesis.

3 Basics

In this chapter we provide a detailed overview of the algorithms and methods used throughout the thesis. First, we present the basic click models and basic offline evaluation metrics that will be used in Chapters 4–8 and Chapters 6–8, respectively. Then we present online evaluation methods used in Chapters 6–8.

3.1 Basic Click Models

Click data has always been an important source of information for search engines. It is, however, an implicit signal—we do not always understand how user behavior correlates with user satisfaction as user's clicks are biased. Following Joachims et al. [92], who conducted eye-tracking experiments, there was a series of papers that model user behavior using *probabilistic graphical model*s (PGMs).[1]

The basic models described below are used throughout the thesis for different purposes. Chapters 4 and 7 use them as baselines. In Chapter 5 the models are the objects that we study and whose evaluation we discuss. In Chapters 6 and 7 we build evaluation metrics on top of click models. Finally, in Chapter 8 we apply click models to simulate users.

A *click model* can be described as follows. Let us assume that a user submits a query q to a search engine and gets back a list of results: $u_1, \ldots, u_n$. Given a query q we denote a *session* to be a set of events experienced by the user since issuing the query until abandoning the result page or issuing another query. Note that one session corresponds to exactly one query. The minimal set of random variables used in all models to describe user behavior are *examination* of the r-th document (E_r) and *click* on the r-th document (C_r):

- E_r indicates whether the user looked at the document[2] at rank r (hidden variables).
- C_r indicates whether the user clicked on the r-th document (observed variables).

Note that for the sake of simplicity we use the rank r as a subscript for the random variables and not the query-document index uq. We are going to use these notations interchangeably, writing either X_{uq}, X_u or X_r depending on the context.

[1] See [102] for a general introduction to PGMs.

[2] Here and below we refer to the snippet as it appears on the SERP, not the full document behind the click.

In order to define a click model we need to denote dependencies between these variables. As there are many ways to do that, there exist a great variety of click models, some of which are presented below.

3.1.1 Random Click Model (RCM)

The *random click model* (RCM) assumes that a document is clicked with probability ρ regardless of document position and its perceived relevance:

$$P(C_r = 1) = \rho. \tag{3.1}$$

This model has only one parameter ρ that is either inferred from the data or set to some value, like, e.g, 0.5.

Two simple extension of this model are:

- *Rank-based CTR model* (RCTR) where click probabilities are different for each rank: $P(C_r = 1) = \rho_r$.
- *Document-based CTR model* (DCTR) where click probabilities are different for each query-document pair: $P(C_u = 1) = \rho_{uq}$.

See [45] for more details.

3.1.2 Position-based Model (PBM)

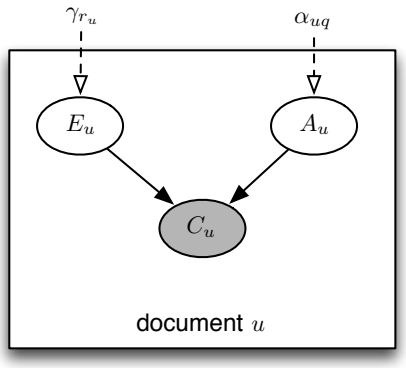


Figure 3.1: Graphical representation of the position-based model.

Many click models include a so-called *examination hypothesis*:

$$C_u = 1 \Leftrightarrow E_u = 1 \text{ and } A_u = 1, \tag{3.2}$$

which means that a user clicks a document u if, and only if, she examined the document ($E_u = 1$) and was attracted by the document ($A_u = 1$). The random variables E_u and A_u are usually considered independent.

The simplest model that uses the examination hypothesis introduces a set of document-dependent parameters α that represent the *attractiveness*[3] of documents on a SERP:

$$P(C_u = 1 \mid E_u = 1) = P(A_u = 1) = \alpha_{uq}. \tag{3.3}$$

We should emphasize that attractiveness here, or probability of click given examination, is a characteristic of the document's snippet (sometimes referred to as document caption), and not the full text of the document. As was shown by Turpin et al. [157], this parameter is correlated with the full-text relevance obtained from judges in a TREC-style assessment process, but there is still a substantial discrepancy between them.

Joachims et al. [92] shows that the probability of a user examining a document depends heavily on its rank or position on a SERP and typically decreases with rank. That was also confirmed by Zhang et al. [167]. To incorporate this intuition into a model, we introduce a set of examination parameters γ_r, one for each rank.[4] This *position-based model* (PBM) was formally introduced by Craswell et al. [59] and can be written as follows:

$$C_u = 1 \Leftrightarrow E_u = 1 \text{ and } A_u = 1 \tag{3.4}$$

$$P(A_u = 1) = \alpha_{uq} \tag{3.5}$$

$$P(E_u = 1) = \gamma_{r_u}. \tag{3.6}$$

The corresponding graphical model is shown in Figure 3.1.

3.1.3 Rank-biased Precision (RBP) Model

If we take PBM (3.4)–(3.6) introduced above and, instead of learning it from data, set the examination probability $P(E_r = 1) = \gamma_r$ to p^{r-1} for some parameter p, we obtain the model behind the rank-biased precision metric [115]. By doing so we reduce the parameter space for γ from $n - 1$ parameters (γ_r for r from 2 to n) to just one parameter p, which, as in case of RCM, can be either learned from the data or set to some fixed value.

3.1.4 Cascade Model (CM)

The *cascade model* (CM) [59] assumes that a user scans documents on a SERP from top to bottom until he or she finds a relevant document. Under this assumption, the top ranked document u_1 is always examined, while documents u_r at ranks $r \geqslant 2$ are examined if, and only if, the previous document u_{r-1} was examined and not clicked. If we combine this idea with the examination assumptions (3.4) and (3.5), we obtain the cascade model

[3] It is sometimes referred to as *perceived relevance* or just *relevance* of a document.

[4] Strictly speaking, in PBM we assume that the first document is always examined, so γ_1 always equals 1, and we therefore have one less parameter.

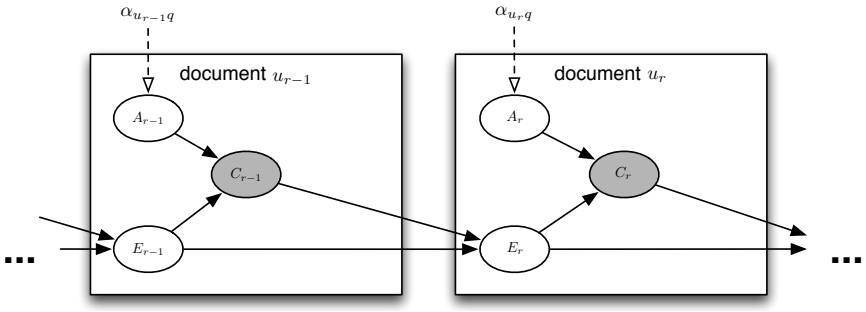


Figure 3.2: Graphical representation of the cascade model (fragment).

as introduced by Craswell et al. [59]:

$$C_r = 1 \Leftrightarrow E_r = 1 \text{ and } A_r = 1 \tag{3.7}$$
$$P(A_r = 1) = \alpha_{u_rq} \tag{3.8}$$
$$P(E_1 = 1) = 1 \tag{3.9}$$
$$P(E_r = 1 \mid E_{r-1} = 0) = 0 \tag{3.10}$$
$$P(E_r = 1 \mid C_{r-1} = 1) = 0 \tag{3.11}$$
$$P(E_r = 1 \mid E_{r-1} = 1, C_{r-1} = 0) = 1. \tag{3.12}$$

The corresponding graphical model is shown in Figure 3.2.

The key difference between the *cascade model* (CM) and the *position-based model* (PBM) is that the probability of clicking on a document u_r in PBM does not depend on the events at previous ranks $r' < r$ whereas in CM it does. In particular, CM does not allow sessions with more than one click, but such sessions are totally possible for PBM.

3.1.5 User Browsing Model (UBM)

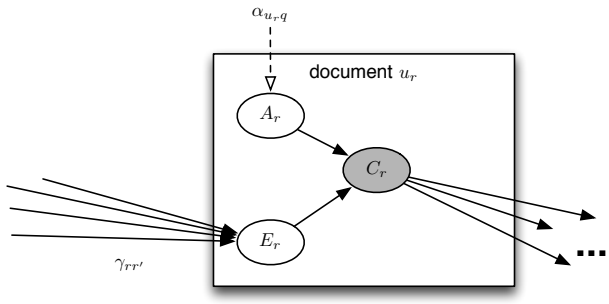


Figure 3.3: Graphical representation of the user browsing model (fragment).

The *user browsing model* (UBM) by Dupret and Piwowarski [69] is an extension of PBM that has some elements of the cascade model. The idea is that the examination probability should take previous clicks into account, but should mainly be position-based. To achieve that, we assume that it depends not only on the rank of a document r, but also on the rank of the previously clicked document r':[5]

$$C_r = 1 \Leftrightarrow E_r = 1 \text{ and } A_r = 1 \tag{3.13}$$

$$P(A_r = 1) = \alpha_{u_r q} \tag{3.14}$$

$$P(E_r = 1 \mid C_1 = c_1, \ldots, C_{r-1} = c_{r-1}) = \gamma_{rr'}, \tag{3.15}$$

where r' is the rank of the previously clicked document or 0 if none of them was clicked. In other words:

$$r' = \max\{k \in \{0, \ldots, r-1\} : c_k = 1\}, \tag{3.16}$$

where c_0 is set to 1 for convenience.

Alternatively, (3.15) can be written as follows:

$$P(E_r = 1 \mid \mathbf{C}_{<r}) = P(E_r = 1 \mid C_{r'} = 1, C_{r'+1} = 0, \ldots, C_{r-1} = 0) = \gamma_{rr'} \tag{3.17}$$

Figure 3.3 shows a graphical representation of UBM. The set of arrows on the left means that the examination probability $P(E_r = 1)$ depends on all click events C_k at smaller ranks $k < r$. Click on rank r, denoted by C_r, in turn, influences all examination probabilities further down the ranked list (hence, the set of arrows on the right side).

3.1.6 Dynamic Bayesian Network (DBN) Model

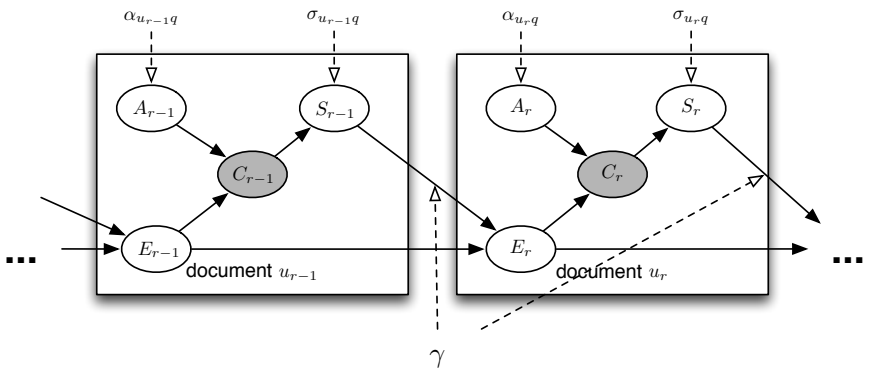


Figure 3.4: Graphical representation of the dynamic Bayesian network model (fragment).

Apart from the examination probability E_r, attractiveness probability A_r and click probability C_r, the *dynamic Bayesian network* (DBN) model [25] also introduces a set of

[5] Instead of the rank of the previously clicked document r', the distance $d = r - r'$ to this document was used in the original paper [69], which is equivalent to the form we use.

random variables S_r, one for each document on a SERP, corresponding to the user's satisfaction after clicking the document at rank r. The DBN model extends CM, and unlike CM it assumes that the user's perseverance after a click depends on the actual relevance[6] σ_{uq}, and not the perceived relevance α_{uq}. In other words, DBN introduces another set of document-dependent parameters. The model, then, can be written as follows:

$$C_r = 1 \Leftrightarrow E_r = 1 \text{ and } A_r = 1 \tag{3.18}$$
$$P(E_1 = 1) = 1 \tag{3.19}$$
$$P(A_r = 1) = \alpha_{u_r q} \tag{3.20}$$
$$P(S_r = 1 \mid C_r = 1) = \sigma_{u_r q} \tag{3.21}$$
$$P(S_r = 1 \mid C_r = 0) = 0 \tag{3.22}$$
$$P(E_r = 1 \mid E_{r-1} = 0) = 0 \tag{3.23}$$
$$P(E_r = 1 \mid S_{r-1} = 1) = 0 \tag{3.24}$$
$$P(E_r = 1 \mid E_{r-1} = 1, S_{r-1} = 0) = \gamma, \tag{3.25}$$

where γ is the continuation probability for a user who either did not click on a document or clicked but was not satisfied by it. The graphical model is shown in Figure 3.4.

A simplification of the DBN model, called *simplified dynamic Bayesian network* (SDBN) is often used. It assumes that $\gamma = 1$ which greatly simplifies the model training and application. As we will see in Chapter 5, its performance is close to the performance of the full DBN.

If we further assume that $\sigma_{u_r q} \equiv 1$ (a click is equivalent to satisfaction), the SDBN reduces to the CM. Another way to reduce SDBN to CM is to set $\alpha_{u_r q}$ to 1, which would give a model that is mathematically equivalent to CM, albeit with different parameter names.

3.1.7 Dependent Click Model (DCM)

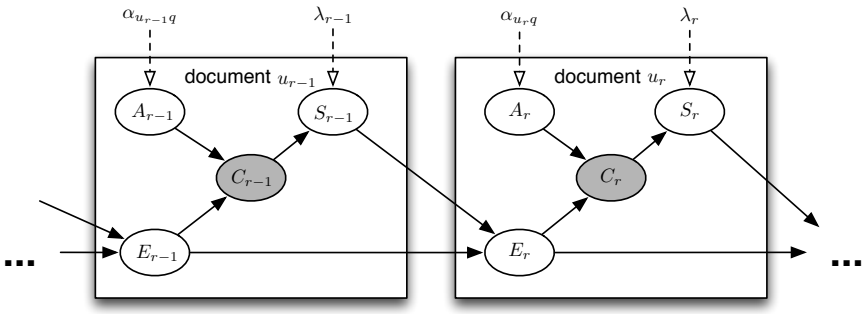


Figure 3.5: Graphical representation of the dependent click model (fragment).

The *dependent click model* (DCM) by Guo et al. [74] is another extension of the cascade model that is meant to handle sessions with multiple clicks. The only difference between

[6]This parameter is often referred to as the *satisfaction probability*.

DCM and SDBN is that the satisfaction probability $P(S_r = 1)$ depends not on the document itself but on its position r in a ranked list. Thus, the DCM model can be described with the following equations (cf. (3.18)–(3.25)):

$$C_r = 1 \Leftrightarrow E_r = 1 \text{ and } A_r = 1 \tag{3.26}$$
$$P(E_1 = 1) = 1 \tag{3.27}$$
$$P(A_r = 1) = \alpha_{u_r q} \tag{3.28}$$
$$P(S_r = 1 \mid C_r = 1) = \lambda_r \tag{3.29}$$
$$P(S_r = 1 \mid C_r = 0) = 0 \tag{3.30}$$
$$P(E_r = 1 \mid E_{r-1} = 0) = 0 \tag{3.31}$$
$$P(E_r = 1 \mid S_{r-1} = 1) = 0 \tag{3.32}$$
$$P(E_r = 1 \mid E_{r-1} = 1, S_{r-1} = 0) = 1, \tag{3.33}$$

The corresponding graphical model is shown in Figure 3.5.

3.2 Basic Offline Metrics

In this section we present the basic metrics used to evaluate IR systems. The general approach, often referred to as the Cranfield evaluation paradigm, assumes there is a collection of query-document pairs that are rated by so-called assessors or raters. For each document u there is a relevance score R_u that quantifies how well the document matches the user query q. The role of an evaluation metric then is to aggregate the scores of all documents in a SERP. Since there are different ways to aggregate relevance scores, we have different evaluation metrics. We call them *offline* metrics, because they can be computed offline and are not based on the users of a live search engine.

Offline metrics presented below are used in multiple chapters. In Chapters 6 and 7 they are used as baselines and compared against. Apart from that, they are used directly as a measurement tool in parts of Chapters 7 and 8.

3.2.1 Precision

The simplest metric used with the binary relevance is the precision-at-n, which tells us how many relevant documents are there in the top-n:

$$\text{Precision} = P@n = \frac{1}{n} \sum_{r=1}^{n} R_r, \tag{3.34}$$

where R_r is the (binary) relevance of the document at rank r. This equation can also be viewed as an average relevance of the documents in the SERP.

3.2.2 Average Precision (AP)

The next step is to approximate the area under the precision-recall curve to get a sense of precision at different recall levels (as opposed to precision at a fixed rank):

$$\text{AP} = \frac{\sum_{r=1}^{n} P@r \cdot R_r}{\sum_{r=1}^{n} R_r}, \tag{3.35}$$

which is equivalent to precision values averaged across ranks of the relevant documents.

3.2.3 Rank-biased Precision (RBP)

As the documents lower in the rank have to be less important, the idea of rank discounting was proposed. One way of discounting has been suggested by Moffat and Zobel [115]:

$$\text{RBP} = \sum_{r=1}^{n} p^{r-1} \cdot R_r, \tag{3.36}$$

where p is the discounting parameter. When p equals 1, the RBP falls back to Precision (3.34). The optimal value of the parameter may be learned from data (see Section 3.1.3) or set to different values to see how it affects the outcome.

3.2.4 Discounted Cumulative Gain (DCG)

Discounted cumulative gain (DCG) is another discounting method introduced by Järvelin and Kekäläinen [89]. Apart from a different rank-dependent discounting factor, it also assumes a graded relevance. It transforms relevance grades[7] R ranging from 0 to R_{max} into real values using the following transformation:

$$\rho(R) = \frac{2^R - 1}{2^{R_{\text{max}}}}. \tag{3.37}$$

The DCG metric can then be written as follows:

$$\text{DCG} = \sum_{r=1}^{n} \frac{\rho(R_r)}{\log_2(1+r)}. \tag{3.38}$$

3.2.5 Normalized Discounted Cumulative Gain (NDCG)

One modification of DCG called *normalized discounted cumulative gain* (NDCG) is somewhat widely used in academia. To compute this metric one first computes the DCG of an ideal SERP where the most relevant documents are put on top and then normalizes on that value:

$$\text{NDCG} = \frac{\text{DCG}}{\text{DCG}_{\text{ideal}}}. \tag{3.39}$$

Since we cannot obtain relevance ratings for all the documents in the world, we cannot compute $\text{DCG}_{\text{ideal}}$ precisely and have to approximate it by the DCG of the SERP made of

[7] In the case of the TREC 2011 Web Track dataset $R_{\text{max}} = 3$, i.e., there are four grades.

the best *rated* documents. Not only is this metric hard to compute, but one may argue that it behaves incorrectly when aggregated across multiple SERPs. For instance, the system that performs well for "hopeless" queries—the queries where $\text{DCG}_{\text{ideal}}$ is low—can in theory have higher average NDCG then the system that performs well for easier queries, where DCG can be brought up to the level useful for the user. On top of that, as was shown by Chapelle et al. [26], the NDCG metric is worse at capturing user satisfaction than DCG.

3.2.6 Cumulative Gain (CG)

There also exists a non-discounted variant of DCG:

$$\text{CG} = \sum_{r=1}^{n} \rho(R_r), \tag{3.40}$$

which is the same as precision in case of binary relevance and differs from it in case of graded relevance.

3.2.7 Expected Reciprocal Rank (ERR)

Finally, another widely used offline metric, *expected reciprocal rank* (ERR), was suggested by Chapelle et al. [26] and is based on the cascade model introduced in Section 3.1.4:

$$\text{ERR} = \sum_{r=1}^{n} \left(\rho_r \prod_{i=1}^{r-1} (1 - \rho_i) \right) \cdot \frac{1}{r}, \tag{3.41}$$

where $\rho_i = \rho(R_i)$ is the same transformation (3.37) as in DCG.

3.3 Basic Online Metrics

A/B-testing, an approach complementary to the Cranfield evaluation paradigm, involves deploying two versions of a system live and comparing various quality metrics based on click signals obtained from the users.

Below we present several commonly used such online quality metrics that we will later use in Chapters 6–8 to compare our evaluation methods against. We give definition for one session even though the metrics are usually averaged for many sessions over a certain period.

- UCTR—binary value representing a click (the opposite of abandonment).
- QCTR (or simply CTR)—number of clicks in a session.
- MaxRR, MinRR, MeanRR—maximum, minimum and mean reciprocal rank ($1/r$) of clicks. Following the work of Radlinski et al. [127] we exclude pages with no clicks when computing these metrics to avoid correlation with UCTR.

- PLC[8]—the number of clicks divided by the rank of the lowest click. This is the opposite of $pSkip$ used in [27].
- Clicks@1—equals 1 if there was a click on the top-1 document, 0 otherwise.

In Chapter 6 these metrics are used as ground truth to show the link between the offline metrics—both the new ones introduced in those chapters and the basic ones discussed in Section 3.2—and the online metrics discussed here. In Chapter 7 we use online metrics to demonstrate a non-trivial dependency between user behavior and the SERP quality. In Chapter 8 we introduce one vertical-specific online metric and then compare all the absolute metrics to pairwise interleaving methods, both classic ones and the new ones that we introduce.

[8] Precision at the lowest click.

4

Click Models for Modern Search Engines

A result page of a modern search engine often contains documents of different types targeted to satisfy different user intents (news, blogs, multimedia). In general, modern SERPs are not just infinite lists of homogeneous results: the results are heterogeneous and the results list, though usually long, is not presented all at once, but is initially truncated to around ten results. All this introduces distortion into user interaction patterns. When evaluating performance of a search system and making design decisions we need to better understand user behavior on such complex result pages. To address this problem various click models have previously been proposed.

In this chapter we introduce a number of modifications to existing click models to improve their performance on complex SERPs. We suggest several ways to improve existing click models to take into account SERP structure and aggregated user information. This corresponds to the following research questions:

RQ1.1 How can we use page layout information to improve click models?

RQ1.2 How can we use aggregated user characteristics such as vertical orientation to improve existing click models?

The following Chapter 5 concerns the questions of evaluating click models, both simple and complex ones. Chapters 6 and 7 discuss an important application of click models—search quality evaluation, while in Chapter 8 we use click models to simulate users.

4.1 Introduction

In this chapter we study two aspects of a search engine result page: the pagination button and vertical search. We show that both of them need to be accounted for while building a click model. The pagination button is a simple element of virtually any search interface, yet it was not previously taken into account by click models. We use it as an example to show how click models can be improved if we start considering such simple user interface elements. Vertical search (Section 2.2), on the other hand, is the de-facto standard of present-day SERPs and therefore we have to design click models with vertical search in mind.

Figure 4.1: Result page switching (pagination) buttons.

A result page of a modern search engine, whether commercial or not, usually has buttons leading to more results; Figure 4.1 shows an example of such buttons. There, the user can switch to the next page of results either by using the page number (e.g., "**2**") or by clicking the "**Next**" button. We have a similar setup in our experiments. By analyzing the 2012 click log of the Yandex search engine we learned that one third of all users uses the pagination buttons at least once a week. At the query level, with probability 5–10%, a user will go to the second result page. This number is even bigger for further result pages—once she has switched to the second page, a user often continues to the third and fourth pages, and this probability is at least five times bigger than the probability of switching from the first to the second page.[1] On average, the users examine 1.1 pages. These facts suggest that we need to pay more attention to the ranking of documents below the first result page—such documents have a non-trivial click pattern and are examined by a substantial number of users. We argue that existing click models do not properly model the click skewness caused by the need for switching pages. We propose a modified DBN click model (see Section 3.1.6 for background), that explicitly includes into the model the probability of transitioning between result pages.

Another distinct feature of contemporary search engines is diversification of results (see Section 2.2). Here we focus on one particular vertical of so-called fresh results—recently published web pages such as news or blogs. Figure 4.2 shows part of a SERP in which fresh results are mixed with ordinary results in response to the query `[chinese islands]`. We say that every document has a *presentation type*, in our example "fresh" (the first two documents in the figure) or "web" (the third, ordinary search result item). We assume that each query has a number of categories—or *intents*—associated with it. In our case these will be "fresh" and "web."

The main problem that we address in this chapter is the problem of modeling user behavior in the presence of pagination buttons and vertical results. In order to better understand user behavior we propose to exploit intent and layout information in a click model to improve its performance. Unlike previous click models our proposed models use additional information that is already available to search engines.

We assume that the search engine already knows the probability distribution of intents corresponding to the query. This is a typical setup for the TREC diversity track [53] as well as for commercial search systems. We also know the presentation type of each document. We argue that this presentation may lead to some sort of bias in user behavior and taking it into account may improve the click model's performance.

Two main contributions of this chapter are the following:

- A modified DBN model, that explicitly includes into the model the probability of transitioning between result pages.

[1] Due to the sensitive nature of this information we cannot disclose the exact numbers.

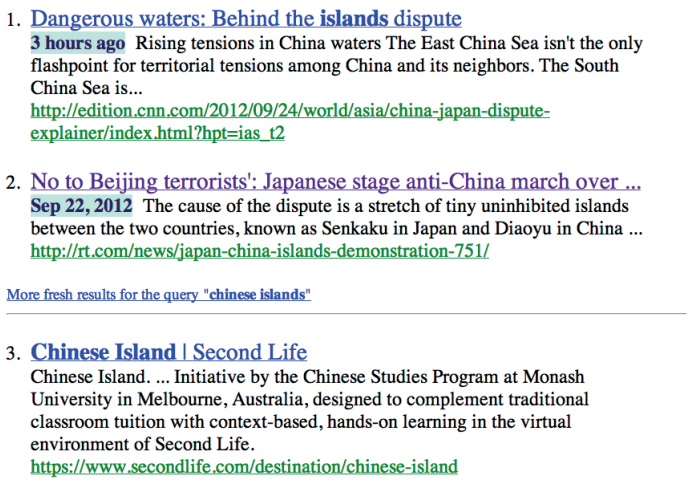


Figure 4.2: Group of fresh results at the top followed by an ordinary web result.

- A novel framework of *intent-aware* (IA) click models that can be used to better understand various aspects of user behavior and document relevance.

In particular, our work on intent-aware click models brings the following contributions: 1) a family of click models that use presentation types of the documents on a SERP and prior knowledge about user intent distribution, 2) a probabilistic framework for dynamic adaptation of this distribution using previous clicks when predicting next clicks, and 3) a breakdown of relevance parameters for different intents, which helps us improve model performance and enables new applications of click models.

In Section 4.2 we introduce our modification of DBN to account for pagination buttons. In Section 4.3 we introduce the family of intent-aware models to account for vertical blocks and user intents. In Section 4.4 we discuss our experimental setup. The results of the experiments are presented in Sections 4.5 and 4.6. Section 4.7 presents an overview of related work. We conclude in Section 4.8.

4.2 Accounting for Layout: Pagination-aware Models

The most straightforward way of extending a click model beyond the first ten result items is to simply apply the model to the entire list of documents, not just the first ten.[2] This model implicitly assumes that the user always clicks a pagination button whenever she is not satisfied/attracted by the current ten documents. Below, we use it as our *baseline* method.

The first model we propose is an extension of the SDBN model described in Section 3.1.6. Essentially, we introduce a fake "document" p corresponding to the pagination

[2]To simplify the discussion we assume that the SERP has exactly ten documents. This simplification can be dropped when implementing the model.

buttons. This fake "document" is shared across all result pages, i.e., model parameters for this document depend only on the user query q. Because this is not a real document we assign a different meaning to the model parameters: the attractiveness α_{u_pq} of the fake "document" p models the probability of proceeding to the next page of results by clicking a pagination button (continuation probability); the satisfaction probability σ_{u_pq} plays the role of an "impatience parameter." The reason to add this "impatience parameter" is that the first document on the second page (u_{11} in our case), once shown, is clicked much less than the first document u_1 on the first page (at least 3 times less on our data). This might be motivated not only by the lower quality of the document, but also by a decrease in trust in the documents after the user has had to click a pagination button, namely, the user abandons search results right after switching to the next result page. This hypothesis also suggests that we cannot simply apply the SDBN model separately to each result page. The resulting model, that we refer to as *simplified dynamic Bayesian network (pagination-aware)* (SDBN(P)), is visualized in Figure 4.3. To simplify the image, we draw it as if a SERP had three documents, even though it has more (usually ten).

So, the SDBN(P) model uses the same equations (3.18)–(3.25) as SDBN. The click probability for the fake "document" representing the pagination buttons is also observed: we have not only changed the model, but also introduced an additional source of information. As in [88], where the authors use information about mouse movements to improve the model, we do not measure how well we predict these additional observed events (pagination clicks)—we are only interested in predicting document clicks and do not include pagination clicks in the perplexity computation (4.7).

We also introduce a second model, SDBN(P-Q), that uses equations (3.18)–(3.25) for normal documents and (3.18), (4.1), (4.2), and (3.22)–(3.25) for the fake "document" p corresponding to the pagination buttons:

$$P(A_p = 1) = \alpha_{u_p} \tag{4.1}$$

$$P(S_p = 1 \mid C_p = 1) = \sigma_{u_p}, \tag{4.2}$$

i.e., we assume that the attractiveness and "satisfaction" parameters do not depend on the query (hence, "minus Q" in our notation SDBN(P-Q)) and are global constants.

4.3 Accounting for Layout and User Intents: Intent-aware Models

Now we address a different part of the SERP layout, namely vertical blocks. Because different vertical blocks are targeted to answer to different user *intents*, we study layout and intent together.

In order to demonstrate how layout and intent information can be used to better understand user behavior we propose modifications to commonly used click models and show that the information added through our modifications helps us improve click model performance. As a basic model to modify we use the user browsing model by Dupret and Piwowarski [69] (Section 3.1.5), although our extensions can equally well be applied to other click models. Unlike Chen et al. [28], we focus on almost-plaintext results that

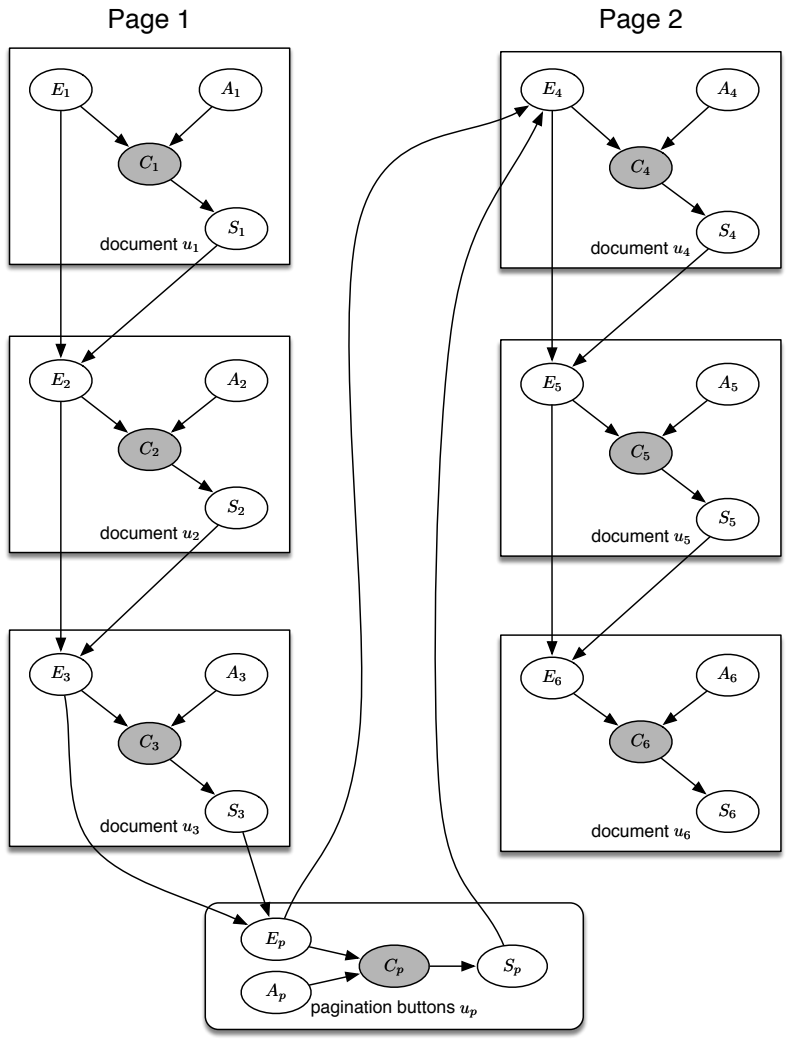


Figure 4.3: A graphical representation of SDBN(P) and SDBN(P-Q); arrows show probabilistic relations. The probabilities of the events A_p and S_p depend on the query in SDBN(P) and do not depend on the query in SDBN(P-Q).

look very similar to the standard "ten blue links." We do not know beforehand if the user notices any differences between special (vertical) results and ordinary ones.

We add one hidden variable I and a set of observed variables $\{G_r\}$ to the two sets of variables $\{E_r\}$ and $\{C_r\}$ commonly used in click models:

- $I = i$ indicates that the user performing the session has *intent* i, i.e., relevance with respect to the category i is much more important for the user.

- $G_r = l$ indicates that the result at rank r has an appearance specific to the results with dominating intent l. For example, for the result page shown in Figure 4.2 we have $G_1 = \mathit{fresh}$, $G_2 = \mathit{fresh}$, $G_3 = \mathit{web}$. We will further refer to a list of presentation types $\{G_1, \ldots, G_{10}\}$ for a current session as a *layout*.

A typical user scenario can be described as follows. First, the user looks at the whole result page and decides whether to examine the r-th document or not. We assume that the examination probability $P(E_r)$ does not depend on the document itself, but depends on the user intent, her previous interaction with other results, the document rank r and the SERP layout. If she decides to examine the document (if $E_r = 1$) we assume that she is *focused* on that particular document. It implies that the attractiveness probability $P(A_r = 1)$ depends only on the user intent I and the attractiveness of the current document, but neither on the layout nor on the document position r. After clicking (or not clicking) the document the user moves to another document following the same "examine-then-click" scenario.

Here we only allow dependencies between E_r and G_r in order to simplify inference, but one can also consider additional dependency links.[3] As an example, using our proposed addition, one can build an intent-aware version of the UBM model in the following manner (cf. (3.13)–(3.15)):

$$C_r = 1 \Leftrightarrow E_r = 1 \text{ and } A_r = 1 \tag{4.3}$$

$$P(A_r = 1 \mid I = i) = \alpha^i_{u_r q} \tag{4.4}$$

$$P(E_r = 1 \mid G_r = b, I = i, C_1, \ldots, C_{r-1}) = \gamma_{rr'}(b, i), \tag{4.5}$$

where α and γ are to be inferred from clicks: $\alpha^i_{u_r q}$ is the attractiveness of the document u_r for the intent i and $\gamma_{rr'}(b, i)$ is the probability of examination given the current rank r, rank of the previous click r', current intent $I = i$ and current presentation type $G_r = b$. The model is shown in Figure 4.4 (cf. Figure 3.3). Below, we refer to (4.3)–(4.5) as *user browsing model (intent-aware)* (UBM-IA).

We would like to stress that our aim here is not to study how to find intents corresponding to the query. Instead, given that we know the query intent spectrum, we aim to investigate the effect of this distribution on the users' click-through behavior. So we assume that for each session we have a prior distribution of the intents $P(I = i)$.[4] Importantly, unlike Hu et al. [85] we do not assume that our intent distribution is fixed for

[3] For example, one can assume that the presentation types of previously clicked documents affects the probability of examination E_r, because they may indicate a bias towards documents of a particular type. See our discussion in Section 4.7 for more details.

[4] We use a proprietary machine-learned algorithm to obtain this value for each query.

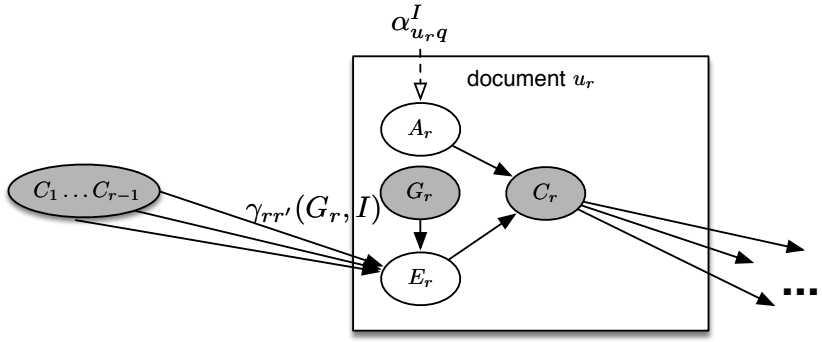


Figure 4.4: The graphical model for UBM-IA. Gray circles correspond to observed variables. Arrows show dependency links.

the session. When predicting the next click, we use posterior intent distribution which we compute using Bayes' rule, see (4.16):

$$P(C_r \mid C_1, \dots, C_{r-1}) = \sum_I \underbrace{P(C_r \mid C_1, \dots, C_{r-1}, I)}_{\text{probability from single intent model}} \cdot \underbrace{P(I \mid C_1, \dots, C_{r-1})}_{\text{posterior intent distribution}}. \quad (4.6)$$

Dupret and Piwowarski [69] found that the single browsing model outperforms a mixture of browsing models when inferring intent distribution from clicks. We show that using layout information and prior knowledge of intent distribution, we can significantly outperform the single browsing model.

4.4 Experimental Setup

In order to test our ideas and answer our research questions, we collect a click log of the Yandex search engine and then use the *maximum likelihood estimation* (MLE) and *expectation maximization* (EM) algorithms to estimate parameters of the click models. To train the pagination-aware models we use the MLE method from [25] (called Algorithm 1 there). Details of the EM algorithm for UBM-IA are described in the Appendix (Section 4.9). Both intent-aware and pagination-aware methods are implemented in our `clickmodels` project.[5]

We use different data samples for testing the pagination-aware and intent-aware models. For the latter, we only select query sessions that have fresh results. We also discard sessions with no clicks (as in [25]) and do not take into account clicks on positions lower than ten.[6] The details of the datasets are summarized in Table 4.1. In both cases we use one day of data for train/test, but for pagination-aware experiments we use a sliding window of two days, whereas for intent-aware we use non-overlapping pairs. For

[5] `https://github.com/varepsilon/clickmodels`.

[6] Fresh results are also counted and might appear at any position.

Table 4.1: Summary of the datasets used to evaluate effects of layout and intent.

Dataset	Dates Collected	Query Sessions	Train / Test Day Pairs
Pagination-aware	2013, Feb 1–11	37,163,170	d_k / d_{k+1}, $k \in 1, \ldots, 11$
Intent-aware	2012, Jul 1–30	14,969,116	d_{2m-1} / d_{2m}, $m \in 1, \ldots, 15$

intent-aware models we have also experimented with the whole dataset (split into equally sized train and test sets) and observed almost identical results.

To evaluate a model on a test set we used the conditional perplexity metric (Section 5.2.3). For each rank r we calculated the following expression:

$$p_r = 2^{-\frac{1}{N}\sum_{j=1}^{N}\left(C_r^j \log_2 q_r^j + (1-C_r^j)\log_2(1-q_r^j)\right)}, \tag{4.7}$$

where C_r^j is a binary value corresponding to the presence of a click on the r-th rank in the j-th session, q_r^j is the predicted probability of a click on the r-th position in the j-th session given the observed previous clicks. Conditional perplexity measures how "surprised" the model is upon observing the click or lack thereof (skip). The higher its value, the worse the model.

The conditional perplexity of a simple random click model—predicting each click with probability 0.5—equals 2, the conditional perplexity of a perfect model is 1. We also report an average perplexity value for ranks from one to ten:

$$AvgPerp = \frac{1}{10}\sum_{r=1}^{10} p_r. \tag{4.8}$$

To compute the perplexity gain of model B over model A we use the following formula:

$$\frac{p_A - p_B}{p_A - 1}, \tag{4.9}$$

which is a standard way to compare perplexity values [28, 45, 74, 169].

4.5 Results: Pagination-aware Models

In this section we compare our pagination-aware SDBN(P) and SDBN(P-Q) models to the original SDBN model. The average conditional perplexity values across ten train/test pairs are reported in Figure 4.5. Vertical bars correspond to the confidence intervals computed using the bootstrap method [71] with 1000 samples and 95% confidence level. In this and other figures we use continuous results numbering, so the first document of the second page has rank $r = 11$.

We can see that the perplexity values of the models do not differ for ranks 1 through 10. However, we do see that the perplexity is much lower for SDBN(P) and SDBN(P-Q) for all positions below rank 10.

Another interesting observation is that we no longer observe constantly decreasing perplexity for $r > 10$. It may well be the case that the underlying cascade hypothesis—that the user examines the documents one by one—needs to be revised for the second and later result pages. We leave this matter as future work.

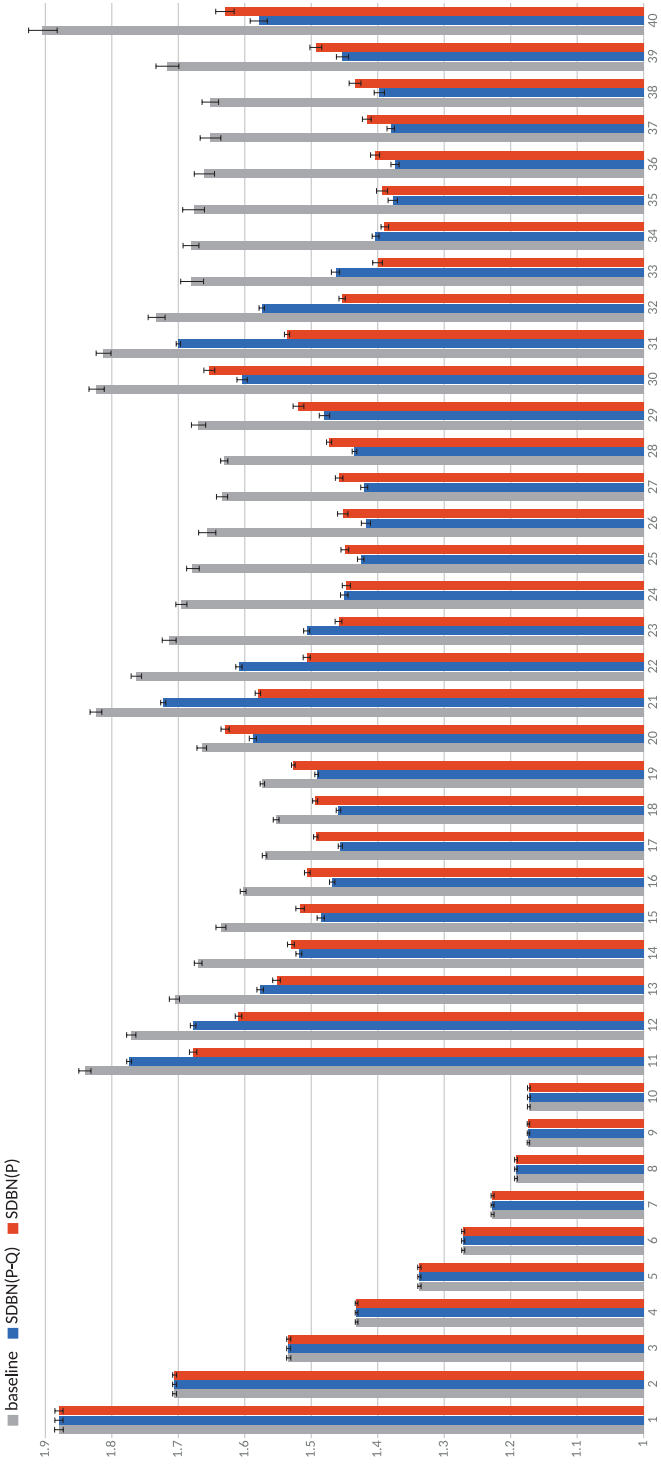


Figure 4.5: Conditional perplexity of click prediction for different models for ranks 1 through 40.

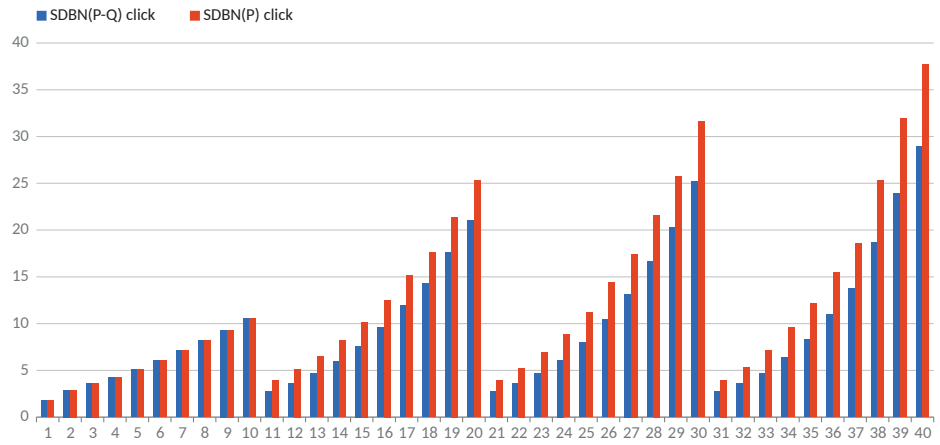


Figure 4.6: Conditional perplexity of predicting click events for SDBN(P) and SDBN(P-Q) (y axis); document rank (x axis).

Next, we contrast the perplexity gains of our two new models over the baseline. Confidence intervals are again computed using the bootstrap method. If we compute the average conditional perplexity values for positions 1 through 40, then the SDBN(P-Q) and SDBN(P) models yield a 20.5% and 20.2% perplexity gain over the baseline, respectively.

Following Dupret and Piwowarski [69], we also analyze the prediction perplexity of clicks and skips (non-clicks) separately. We compute p_r^{click} and p_r^{skip} using equation 4.7 limiting ourself to C_r^j equal to 1 and 0 respectively. The corresponding results are reported in Figures 4.6 and 4.7. Both models have difficulties predicting clicks, but since click events occur less often than skips, the overall perplexity is not very high. From these plots we can see that SDBN(P) is always better at predicting skips and worse at predicting clicks.

Returning to Figure 4.5, we can see that on every result page for ranks 20–40, SDBN(P) outperforms SDBN(P-Q) at the beginning of the result page, while SDBN(P-Q) wins at the end of the result page. We believe that the reason is that the relative importance of click/skip prediction is different at the top and the bottom of a result page.[7]

4.6 Results: Intent-aware Models

In the previous section we have demonstrated that accounting for layout elements such as pagination buttons boosts the predictive abilities of the model substantially. Now we turn our attention to another layout element, namely presentation type of vertical results. We also consider the notion of user intent distribution, which is closely related to the vertical result type.

[7] Recall that SDBN(P-Q) is better at predicting clicks while SDBN(P) better predicts absence of clicks.

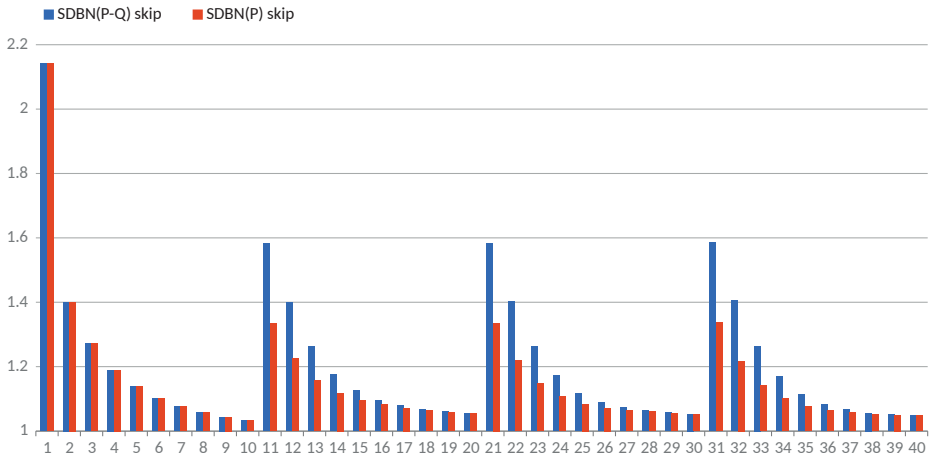


Figure 4.7: Conditional perplexity of predicting non-click events for SDBN(P) and SDBN(P-Q) (y axis); document rank (x axis).

Table 4.2: Average perplexity gain for UBM-IA.

Model	Average Perplexity Gain	Confidence Interval (Bootstrap)
UBM-IA vs. UBM	1.34 %	$[1.25\%, 1.43\%]$

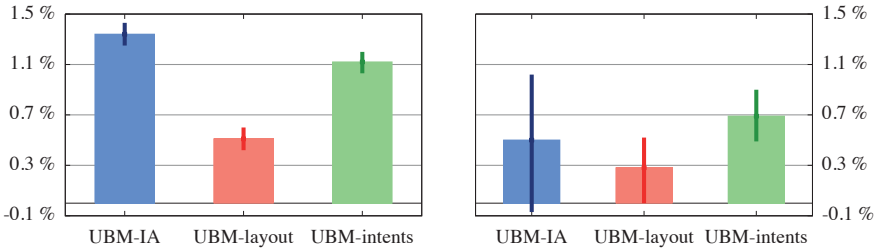
As a starting point we implemented the classical DBN and UBM models[8] and tested them on our data. We found that UBM performs much better than DBN, consistently giving around 18% gain in perplexity over DBN. So we decided to use UBM as our baseline and we report our improvements compared to this model.

4.6.1 Layout and Intent Information

The combined contribution of layout and intent. We start by comparing our UBM-IA model (4.3)–(4.5) to the original UBM model and then consider the individual contributions of intent and layout information. The main results are summarized in Table 4.2. In this table we report the average value of perplexity gain for the 15 subset pairs (d_{2m-1}, d_{2m}) described in Section 4.4 (Table 4.1). We also report confidence intervals calculated using the bootstrap method [71]. We can see that our improvements are statistically significant.

Layout and intent in isolation. When we take a look at the modifications implemented on top of the UBM model, (4.3)–(4.5), we can see that they are actually a combination of

[8]See Section 3.1.



(a) Fresh results with special snippets

(b) Fresh results with ordinary snippets

Figure 4.8: Perplexity gains for UBM-IA, UBM-layout and UBM-intents compared to UBM.

two ideas: information about layout $\{G_r\}$ and information about user intents I. These ideas can be tested separately and we can study their contribution separately. We call the resulting click models *user browsing model (vertical layout)* (UBM-layout) and *user browsing model (intents)* (UBM-intents); they are defined using (4.3), (4.10), (4.11) and (4.3), (4.4), (4.12), respectively:

$$P(A_r = 1) = \alpha_{u_r q} \tag{4.10}$$

$$P(E_r = 1 \mid G_r = b, C_1, \ldots, C_{r-1}) = \gamma_{rr'}(b) \tag{4.11}$$

$$P(E_r = 1 \mid I = i, C_1, \ldots, C_{r-1}) = \gamma_{rr'}(i). \tag{4.12}$$

The results, in terms of perplexity, of comparing UBM-IA, UBM-layout and UBM-intents against UBM are summarized in Figure 4.8(a). We can see that both individual models give some improvements while the best results are achieved using the combined model UBM-IA. Using the bootstrap method we confirm that the observed differences are statistically significant; the confidence intervals are shown as vertical bars.

The importance of layout information. How much of the positive effects observed in Figure 4.8(a) is due to layout information, that is, to the fact that fresh results are singled out and clearly presented as such? In order to answer this question we performed the following user experiment. A small part of all Yandex users were presented with fresh results that looked just like ordinary documents while placed on the same positions. In other words, despite the fact that the search engine knows the presentation type G_r of every document, these users could not see it. We hypothesize that the usage of layout information will be less reliable in this situation because users with *fresh* intent are less inclined to examine these documents. We collected the data for a period of 12 days in September 2012 and evaluated the same three click models (UBM-IA, UBM-layout and UBM-intents) on this data.

The results, again in terms of perplexity gain, are shown in Figure 4.8(b). Because we have much less data (121,431 sessions corresponding to 42,049 unique queries) our bootstrap confidence intervals are wide, wider than in Figure 4.8(a). From the plot we see that including layout information does not help, and that the best model in this situation is UBM-intents, which affirms our hypothesis.

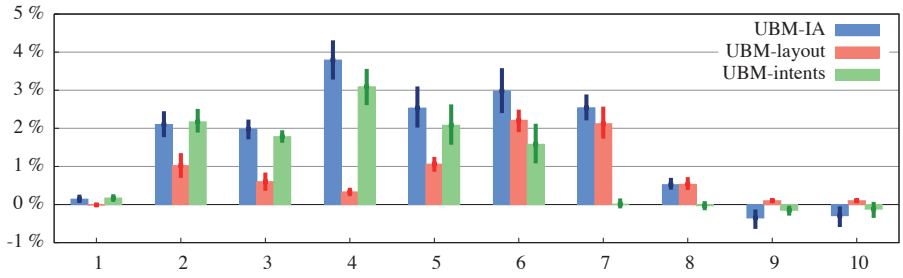


Figure 4.9: Perplexity gains for different ranking positions compared to UBM.

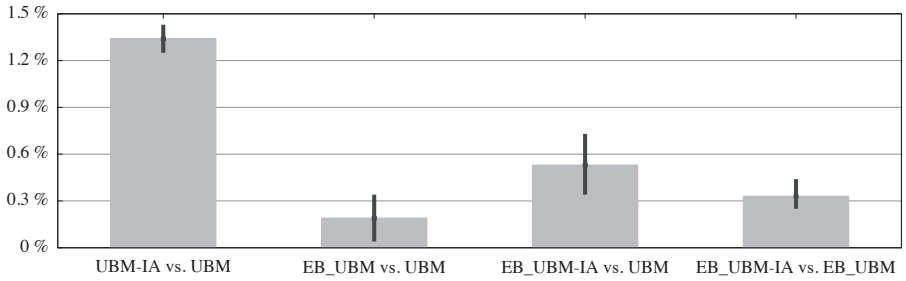


Figure 4.10: Perplexity gains for different models.

Gain per rank. The results so far report on perplexity gains over the complete SERP. We now examine the perplexity gains per individual rank to analyze our click models in more detail. Figure 4.9 shows the results for all three models: UBM-IA, UBM-layout and UBM-intents. One can see that it is difficult to make an improvement for the first document. This is because the models do not differ much for the first position: users usually examine the first document despite its presentation type and other factors, and, therefore, click probability is determined only by the attractiveness of the document. Clicks on the last two positions do not appear to be motivated by user intent or page layout; this information even leads to a decrease in perplexity for the UBM-intents and UBM-IA click models. However, UBM-layout is robust to such errors: it always gives an improvement even if it is mostly smaller than that of other models.

There is another interesting observation to be made. Intent information matters for positions 2–6, while layout information matters for positions 2–10, and it is more important than intent for positions 6–10. This difference can be explained by the fact that for most of the users only the top five or six documents can be viewed without scrolling.

4.6.2 Other Models

We also implement the DBN [25], DCM [75] and RBP [115] click models (Section 3.1). Since these models all performed significantly worse than UBM on our data, they are also expected to perform worse than our UBM-IA click model.

As we mentioned previously, Chen et al. [28] also addressed the problem of verticals by a click model. We can consider their model as a state-of-the-art click model for diversified search. While the focus of that work was on visually appealing verticals, the ones containing images or video, we can assume that our fresh results are similar to their News vertical. We use the best performing click model for that vertical called "exploration bias model" that was based on UBM. Here, we refer to it as *exploration bias user browsing model* (EB_UBM).

Our *exploration bias user browsing model (intent-aware)* (EB_UBM-IA) extends EB_UBM in the same way as we do it for UBM. We compare UBM, UBM-IA, EB_UBM and EB_UBM-IA on our data set. A short summary is reported in Figure 4.10. We see that UBM-IA gives a bigger improvement over the original UBM model than EB_UBM. We can also see that if we combine our ideas of layout and intent with Chen et al.'s idea of "exploration bias," thus yielding the EB_UBM-IA click model, we observe a gain over EB_UBM but not as big as for UBM-IA. That means that there is a certain overlap between those two approaches, yet there is value in combining them.

4.7 Discussion and Related Work

The problem we address in this chapter is closely related to the vertical search problem discussed in Section 2.2. Historically, there are two different approaches to this problem. We can call them the *intent* and *vertical* approaches.

The *intent* hypothesis assumes that each document has separate relevances for different user intents.[9] Following this hypothesis, a family of diversity-aware metrics such as ERR-IA or α-NDCG emerged (see Section 2.4). There were also attempts to use intent information in click models. In the original publication on UBM by Dupret and Piwowarski [69], a so-called *mixture model* was studied. Instead of using prior knowledge of intent distribution they learned such information from clicks. However, they were unable to report any improvements compared to the vanilla UBM (referred to as the *single browsing model* there). In a later publication, Hu et al. [85] proposed to use a constant relevance discount factor for each session to model intent bias. While their approach is valid for building a click model, the variable they used to implicitly model intent bias does not correspond to the commonly used notion of *intents* or *categories* to which we adhere in the current work.

Vertical or *federated* search is an approach adopted by many modern search engines. Following this approach, an incoming query is sent to several specialized search engines, called *verticals*, e.g., Image, Video or News vertical. The returned heterogeneous documents are then organized on a SERP by inserting vertical documents into fixed positions between the organic web results. Relevant related work is discussed in Section 2.2. On top of what is discussed there, particularly relevant to the research in the current chapter is the work on click models [28, 159] that also follows the vertical approach.

In [5], where the problem of vertical selection is studied in detail, there is a list of commonly used verticals such as News, Image, Video, TV, Sports, Maps, Finance. Most of these contain images or interactive tools like video or maps. On the one hand, the fact that we focused on fresh results here can be viewed as a limitation of our work. On the

[9] Frequently referred to as *categories*, *topics* or *nuggets*; see, e.g., Clarke et al. [52].

other hand, there are many user intents other than news, that can be (and probably should be) covered by more or less textual results: forums, blogs, reviews, etc.

Seo et al. [142] studied a problem that is related to the pagination-related skewness problem we address in this chapter. They study a similar problem in the context of aggregated search interfaces. In their setup, the search engine returns a set of vertical blocks, each having a so-called *rank cut*—only the top five documents from the vertical are shown. They showed that users rarely continue to a vertical-specific result page, switching to another vertical instead. Hence, they observe skewness in the number of clicks received by a particular vertical. To deal with this skewness the authors propose to use regularization of the click counts. Our pagination-aware approach differs in that we adopt the framework of click models [45].

An early study by Dumais et al. [68] suggested that users tend to prefer grouped results as they are easier to investigate. It takes less time for participants of their experiment to complete search tasks using a grouped interface. However, if we build a SERP so as to optimize a diversity metric (e.g., ERR-IA) we will end up with a blended result page where results from the same vertical are not necessarily grouped together. To address this problem we ran an online A/B-testing experiment where some users were presented with fresh results grouped while other users always saw fresh results mixed with ordinary web results. We found that fresh results got 5% fewer clicks when they are mixed with other results while the total number of clicks and abandonments remained unchanged.[10] This suggests that if we want to optimize traffic on fresh results (e.g., if news content providers share some revenue with the web search company), we need to consider the fact that user behavior depends on how we organize vertical results. One can extend our intent-aware click model framework to handle these types of layout changes by introducing additional dependencies between the examination probability E_r and the page layout $\{G_r\}_{r=1}^{n}$. For example, for our UBM-IA click model we can add a dependency on the number of vertical groups or presentation type of the previous document G_{r-1} to the $\gamma_{rr'}$ function (4.5).

4.8 Conclusion

In this section we look back at the research questions formulated at the beginning of the chapter and show how we answered them.

The first question runs as follows:

RQ1.1 How can we use page layout information to improve click models?

To answer this question we showed how taking into account two separate layout aspects of a SERP can help us to build better models.

First, we studied the skewness of clicks due to SERP pagination. We showed that the click pattern for documents beyond the first page is different from that of the first page. We introduced new ingredients to the widely used DBN model and showed that by explicitly adding pagination buttons into our model we can achieve better results in predicting clicks beyond the first result page.

Second, we introduced a framework of intent-aware click models, which incorporates layout and intent information. We showed that using presentation type of documents helps

[10] The difference is significant at level $\alpha = 0.001$ when using a two-tailed Mann–Whitney U test.

to improve the performance of click models. Another important property of intent-aware additions to click models is that by analyzing examination probabilities (e.g., $\gamma_{rr'}$ in the case of UBM) we can see how user patience depends on his/her intent and SERP layout. Put differently, it allows us to use a click model as an ad-hoc analytic tool.

Apart from that, intent-aware models helped us to answer the second question:

RQ1.2 How can we use aggregated user characteristics such as vertical orientation to improve existing click models?

We showed that using separate relevance parameters for different user intents helps to improve the model. Our intent-aware modification can be applied to any click model to improve its perplexity. One interesting feature of an intent aware click model is that it allows us to infer separate relevances for different intents from clicks. These relevances can be further used as features for specific vertical ranking functions, e.g., one can tune image search ranking by observing users' clicks on an aggregated SERP that has image block among other results.

Future directions. One limitation of our pagination buttons analysis is that we focused our attention on a cascade-like DBN model. While the ranks of user clicks generally appear in ascending order (Chapelle and Zhang [25] showed that only 3% of the query sessions contain out-of-order clicks in the top ten), out-of-order clicks are more prominently visible when analyzing pagination buttons—users often skip the next result page or return to the previously viewed page. While we simply ignored such sessions (13% of sessions in our sample), it might be interesting to analyze such sessions in detail.

One direction for future work is to consider SERP elements other than pagination buttons or special result pages used in mobile and tablet search. The mobile version of a search result page usually has a different design from the desktop version of the result page so as to facilitate user interaction. In particular, the pagination may be invoked differently: sometimes the new set of results appears after scrolling to the bottom of the screen and no buttons have to be clicked. Vertical blocks also have a different presentation on mobile search, sometimes requiring a user to do an extra tap or swipe to get more information. It would be interesting to check whether our models introduced in this chapter are applicable to mobile and tablet search interfaces.

Another future direction is an analysis of different query classes and different users. By having some prior information about the query and the user we can further refine the model by adjusting the probabilities related to the pagination buttons, e.g., by using different priors for different query classes or different groups of users. We can even go further and make all model parameters, e.g., α and σ in DBN (3.18)–(3.25), depend on the user.

Apart from classes of queries and users, different classes of verticals need to be analyzed. While we only looked at the Fresh vertical here, other verticals may pose different challenges. We performed a set of preliminary experiments using Mobile Applications as a vertical. A result item from this vertical consists of a text snippet with a small thumbnail, price and application rating (see Figure 8.4). These documents are more visually appealing than fresh results but still look similar to web results (unlike video or images). The data was collected during several days in September 2012 and consisted of $34{,}917$ sessions. We found that both UBM-IA and EB_UBM-IA give an improvement of

about 9% perplexity over UBM, while EB_UBM without our modifications gives only a 0.15% improvement. It would be interesting to perform a full-scale study of the model performance for different verticals as a future work.

Sometimes, intents are unique to the query, like for instance for the query `[jaguar]` there are at least two intents: finding information about cars and finding information about animals. It is very unlikely that a search engine has a special vertical for these intents. However, we believe that knowledge of the user's intent can still be used in order to better understand her behavior. Applying our ideas to these minor intents is an interesting direction for future work.

In Chapter 5 we discuss evaluation of click models, both simple ones and vertical-aware. It is recommended to at least skim over this chapter before proceeding to the next ones to have a full picture of the click models.

4.9 Appendix: EM for UBM-IA Model

We describe an expectation maximization algorithm for the UBM-IA click model. Algorithms for the other intent-aware models considered in the chapter can be derived in a similar manner.

Suppose that we have N sessions and a record of URLs shown, their visual representations G_r and click positions. Let us denote the vectors of observed variables as C^j and G^j and the vectors of hidden variables as E^j and A^j. Each vector has length 10 indexed by subscript, e.g., C_r^j is a binary variable denoting whether the r-th document was clicked in the j-th session. We use I^j to denote a hidden variable representing the intent for the session.

M-step. At the M-step we estimate the vector of parameters θ from the previous estimation θ^t:

$$\theta^{t+1} = \arg\max_{\theta} \sum_{y} P\left(Y = y \mid X, \theta^t\right) \log P(X, Y \mid \theta), \tag{4.13}$$

where X and Y denote the sets of observed and hidden variables respectively. In our case:

$$\begin{aligned}
\alpha_{uq}^i &= \arg\max_{\alpha} \sum_{j=1}^{N} \sum_{r=1}^{10} \mathcal{I}(u_r^j = u) \left(q_{A_r}(0, i) \log(1-\alpha) + q_{A_r}(1, i) \log \alpha\right) + \\
&\quad \log P(\alpha) \\
\gamma_{rr'}(b, i) &= \arg\max_{\gamma} \sum_{j=1}^{N} \mathcal{I}(G_r^j = b, \mathit{PrevClick} = r') \cdot \\
&\quad \left(q_{E_r}(0, i) \log(1-\gamma) + q_{E_r}(1, i) \log \gamma\right) + \log P(\gamma),
\end{aligned}$$

where $P(\alpha)$, $P(\gamma)$ are beta priors and q_{A_r}, q_{E_r} are calculated during the E-step.

E-step. Let us first define the probabilities we need to compute:[11]

$$q_{A_r}(a, i) = P(A_r = a, I = i \mid C, G) \tag{4.14}$$
$$q_{E_r}(e, i) = P(E_r = e, I = i \mid C, G). \tag{4.15}$$

We can transform (4.14) and (4.15) using Bayes' rule. E.g., for A_r we have:

$$P(A_r, I \mid C, G) = P(A_r \mid I, C, G) \cdot P(I \mid C, G).$$

The probability $P(I \mid C, G)$ can be calculated as follows:

$$P(I \mid C, G) = \frac{P(C \mid I, G)P(I)}{\sum_{i'} P(C \mid I = i', G)P(I = i')}, \tag{4.16}$$

where $P(I)$ is a prior distribution of intents for a query (assumed to be known). Now, if $C_r = 0$:

$$\begin{aligned} P(A_r = 1 \mid I = i, C, G) &= \frac{\alpha^i_{uq}\,(1 - \gamma_{rr'}(b, i))}{1 - \alpha^i_{uq}\gamma_{rr'}(b, i)} \\ P(E_r = 1 \mid I = i, C, G) &= \frac{\gamma_{rr'}(b, i)(1 - \alpha^i_{uq})}{1 - \alpha^i_{uq}\gamma_{rr'}(b, i)}. \end{aligned}$$

If $C_r = 1$ then $P(A_r = 1 \mid I = i, C, G) = 1$ and $P(E_r = 1 \mid I = i, C, G) = 1$. By combining these equations with (4.16) we complete the E-step.

[11] We omit the superscript j here for convenience.

5 Evaluating Click Models

In this chapter we turn our attention to the question of evaluating click models, including basic models and models accounting for the heterogeneous nature of modern SERPs. We ask ourselves the following research questions:

RQ2.1 How do different click models perform when evaluated on a common dataset?

RQ2.2 How should we evaluate click models for complex aggregated SERPs?

5.1 Introduction

We start with a brief overview of basic click model evaluation methods and present a comprehensive comparison of existing click models using a publicly available dataset and an open-source implementation of click models. This is needed in order to eliminate a prominent drawback of existing studies that usually evaluate only a subset of available click models, often use proprietary data, rely on different evaluation metrics and rarely publish the source code used to produce the results. All these make it hard to compare results reported in different studies.

We then turn our attention to click models for complex SERPs. In addition to the classic evaluation methods that we introduce in the first part, we also propose to look at the model performance from a different angle. Our new evaluation method designed specifically for modern heterogeneous SERPs allows us to isolate model components and therefore gives a multi-faceted view on a model's performance.

5.2 Basic Click Model Evaluation

In Section 4.4 we already mentioned a commonly used conditional perplexity metric (4.7). Below we discuss other commonly used evaluation metrics.

5.2.1 Log-likelihood

Whenever we have a statistical model, we can evaluate its accuracy by looking at the likelihood of some held-out test set (see, e.g., [72]). For each session s in the test set $\mathcal{S}$

we compute how likely this session is according to our click model:

$$\mathcal{L}(s) = P_M\left(C_1 = c_1^{(s)}, \ldots, C_n = c_n^{(s)}\right), \tag{5.1}$$

where P_M is the probability measure induced by the click model M. If we further assume independence of sessions, we can compute the logarithm of the joint likelihood:

$$\mathcal{LL}(M) = \sum_{s \in \mathcal{S}} \log P_M\left(C_1 = c_1^{(s)}, \ldots, C_n = c_n^{(s)}\right). \tag{5.2}$$

This metric is known as *log-likelihood* and usually computed using the formula of total probability:

$$\mathcal{LL}(M) = \sum_{s \in \mathcal{S}} \sum_{r=1}^{n} \log P_M\left(C_r = c_r^{(s)} \mid \mathbf{C}_{<r} = \mathbf{c}_{<r}^{(s)}\right). \tag{5.3}$$

As a logarithm of a probability measure, this metric always has non-positive values with higher values representing better prediction quality. The log-likelihood of a perfect prediction equals 0.

5.2.2 Perplexity

Craswell et al. [59] proposed to use the notion of *cross-entropy* from information theory [117] as a metric for click models. This metric was not easy to interpret and it did not become widely used. Instead, Dupret and Piwowarski [69] proposed to use the conceptually similar notion of *perplexity* as a metric for assessing click models:

$$p_r(M) = 2^{-\frac{1}{|\mathcal{S}|}\sum_{s \in \mathcal{S}}\left(c_r^{(s)} \log_2 q_r^{(s)} + \left(1 - c_r^{(s)}\right) \log_2\left(1 - q_r^{(s)}\right)\right)}, \tag{5.4}$$

where $q_r^{(s)}$ is the probability of a user clicking the document at rank r in the session s as predicted by the model M, i.e., $q_r^{(s)} = P_M(C_r = 1 \mid q, \mathbf{u})$. Note that while computing this probability only the query q and the vector of documents $\mathbf{u}$ are used and not the clicks in the session s. This is different from the conditional perplexity definition (4.7) that we used before.

Usually, perplexity is reported for each rank r independently, but it is also often averaged across ranks:

$$p(M) = \frac{1}{n} \sum_{r=1}^{n} p_r(M). \tag{5.5}$$

Notice that perplexity of the ideal prediction is 1. Indeed, an ideal model would predict the click probability q_r to be always equal to the actual click c_r (zero or one). On the other hand, the perplexity of the simple random model where each document is clicked with probability $\rho = 0.5$ is equal to 2. Hence, the perplexity of a realistic model should lie between 1 and 2. Notice, also, that better models have lower values of perplexity. When we compare perplexity scores of two models A and B, we follow Guo et al. [75] and compute the *perplexity gain* of A over B as follows:

$$gain(A, B) = \frac{p^B - p^A}{p^B - 1}. \tag{5.6}$$

The perplexity is typically higher for top documents and decreases toward the bottom of a SERP.[1]

5.2.3 Conditional Perplexity

One of the building blocks of perplexity is the click probability q_r at rank r. In the above definition we defined it as the full probability $P(C_r = 1)$. This probability is independent of the clicks in the current session s and could be tricky to compute as it involves marginalizing over all possible assignments for previous clicks $\mathbf{C}_{<r}$.

Another possibility for defining perplexity is to use the conditional click probability, given the previous clicks in the session. We introduce another version of perplexity, which we call *conditional perplexity*:

$$\tilde{p}_r(M) = 2^{-\frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}\left(c_r^{(s)}\log_2\tilde{q}_r^{(s)}+\left(1-c_r^{(s)}\right)\log_2\left(1-\tilde{q}_r^{(s)}\right)\right)}, \tag{5.7}$$

where $\tilde{q}_r^{(s)}$ is the conditional probability:

$$\tilde{q}_r^{(s)} = P_M\left(C_r = 1 \mid C_1 = c_1^{(s)}, \dots C_{r-1} = c_{r-1}^{(s)}\right). \tag{5.8}$$

Unlike the simple version of perplexity (5.4), which evaluates a model's ability to predict clicks without any input, the conditional perplexity is just a decomposition of the likelihood (Section 5.2.1) and also factors in a model's ability to exploit clicks above the current rank. Since the conditional perplexity uses the beginning of a session $\mathbf{c}_{<r}^{(s)}$ to compute the probability $\tilde{q}_r^{(s)}$ it typically yields lower values than the simple notion of perplexity presented above, especially for models like UBM (Section 3.1.5) that rely on the accuracy of previous clicks. It is important to keep this in mind when comparing results from different publications using different versions of perplexity.

5.3 Evaluating Vertical-aware Models: Intuitiveness

While all the metrics described above are applicable to complex SERPs, we would like to have a better insight into the performance of the vertical-aware click models—the models that take into account the groups of vertical results, such as the models we introduced in Chapter 4.

Sakai [132] proposed a way of quantifying "which metric is more intuitive." This method has been applied to understanding aggregated search metrics in [174], where four key factors of aggregated search systems are listed: *vertical selection* (VS), *item selection* (IS), *result presentation* (RP), *vertical diversity* (VD). The authors measure the preference agreement of a given aggregated search metric with a "basic" single-component metric for each factor; they also assess the ability of a metric to capture the combination of these factors.

The main contribution of this section is that we adapt the intuitiveness test to evaluate vertical-aware *click models* instead of aggregated search metrics. In order to apply the intuitiveness test to click models, we use a simulation setup and proceed as follows. We

[1] As we have shown in Section 4.5, this does not hold beyond the first SERP.

run a click model CM to simulate user clicks[2] and report the total number of clicks—CTR—produced by the simulated user as a metric score for a given ranking. We then compare aggregated search systems by the number of clicks they receive according to a click model CM, like in A/B-testing experiments.[3] The outcome of this aggregated search system comparison determines the intuitiveness of the underlying click model.

Algorithm 5.1 shows our intuitiveness test algorithm. The algorithm computes relative intuitiveness scores for a pair of click models CM_1 and CM_2 and a gold standard metric M_{GS}. The latter represents a basic property that a candidate metric should satisfy. We consider not one but four metrics as our gold standards, one for each aggregated search factor; the same metrics were used by Zhou et al. [174]. These gold standards are intentionally kept simple and some of them are set retrieval metrics based on binary relevance. They should be agnostic to differences across models (e.g., different position-based discounts); their purpose is to separate out and test single factor properties of more complex click models. The four gold standard metrics are:

- VS metric—selected vertical precision on relevant verticals with high vertical orientation (orientation values more than 0.5).
- VD metric—selected vertical recall on all available verticals.
- IS metric—mean precision of retrieved vertical documents.
- RP metric—Spearman's rank correlation with a "perfect" aggregated search reference page.

We first obtain all pairs of aggregated search result pages for which CM_1 and CM_2 disagree about which result page should get more clicks. Out of these disagreements, we count how often each click model's CTR scores agree with the gold standard metric(s). The click model that concords more with the gold standard metric(s) is considered to be more "intuitive." An ideal click model should be consistent with all four gold standards; we therefore introduce an additional step to Algorithm 5.1 where we count how often the model agrees with a subset or all of the four gold standards at the same time.

When compared to traditional perplexity-based click model evaluation as presented above, our method has the following advantages:

- it allows for assessments of *individual* model components, separating their contribution to the model's performance;
- it assigns *explanatory* scores that allow us to assess the ideas underlying a click model;
- it allows us to make use of public test collections and obtain *re-usable* scores without the need to access a user click log.

[2] The code we use to simulate clicks is available as part of Lerot [137] at `https://bitbucket.org/ilps/lerot`.

[3] Alternatively, one can perform an interleaving comparison (see Section 2.5), but this is beyond the scope of the current experiment.

Algorithm 5.1 Computing the intuitiveness scores of click models CM_1 and CM_2 based on preference agreement with a gold standard metric M_{GS}.

$Disagree \leftarrow 0$; $Correct_1 \leftarrow 0$; $Correct_2 \leftarrow 0$
for all pairs of runs (r_1, r_2) **do**
  for all TREC topic t **do**
    $\delta_1 \leftarrow CTR_{CM_1}(t, r_1) - CTR_{CM_1}(t, r_2)$
    $\delta_2 \leftarrow CTR_{CM_2}(t, r_1) - CTR_{CM_2}(t, r_2)$
    $\delta_{GS} \leftarrow M_{GS}(t, r_1) - M_{GS}(t, r_2)$
    if $\delta_1 \times \delta_2 < 0$ **then** ▷ CM_1 and CM_2 disagree
      $Disagree \leftarrow Disagree + 1$
      if $\delta_1 \times \delta_{GS} \geqslant 0$ **then** ▷ CM_1 and M_{GS} agree
        $Correct_1 \leftarrow Correct_1 + 1$
      if $\delta_2 \times \delta_{GS} \geqslant 0$ **then** ▷ CM_2 and M_{GS} agree
        $Correct_2 \leftarrow Correct_2 + 1$
$Intuitiveness(CM_1 \mid CM_2, M_{GS}) \leftarrow Correct_1 \,/\, Disagree$
$Intuitiveness(CM_2 \mid CM_1, M_{GS}) \leftarrow Correct_2 \,/\, Disagree$

5.4 Basic Evaluation: Experimental Comparison

In this section, we present a comprehensive evaluation of the click models described in Section 3.1 using a publicly available dataset, an open-source implementation and a set of commonly used evaluation metrics introduced in Section 5.2.

5.4.1 Experimental Setup

We use the WSCD 2012 dataset released by Yandex as part of the Web Search and Click Data workshop. We use the first one million query sessions of the dataset to train and test the basic click models.[4] We train models on the first 75% of the sessions and test them on the last 25%. This is done to simulate the real-world application where the model is trained using historical data and applied to unseen future sessions. Since sessions in the dataset are grouped by user tasks, we are not fully guaranteed to have a strict chronological ordering between training and test material, but this is the closest we can get. Since the basic click models introduced in Section 3.1 cannot handle unseen query-document pairs, we filter the test set to contain only queries that appear in the training set.

In order to verify that the results hold for other subsets of the data and to report confidence intervals for our results, we repeat our experiment 15 times, each time selecting the next million sessions from the dataset and creating the same training-test split as described above. We then report the 95% confidence intervals using the bootstrap method [71].

All click models are implemented in Python within the PyClick library.[5] The *expectation maximization* (EM) algorithm uses 50 iterations. The DCM model is implemented in its simplified form, *simplified dependent click model* (SDCM), according to the original

[4] We also experimented with ten million sessions and the results were qualitatively similar to what we present here.

[5] `https://github.com/markovi/PyClick`.

Table 5.1: Log-likelihood, perplexity, conditional perplexity and training time of basic click models for web search as calculated on the first one million sessions of the WSCD 2012 dataset. The best values for each metric are indicated in boldface.

model	log-likelihood	perplexity	conditional perplexity	time (sec)
RCM	−0.3727	1.5325	1.5325	**2.37**
RCTR	−0.3017	1.3730	1.3730	2.45
DCTR	−0.3082	1.3713	1.3713	9.39
PBM	−0.2757	1.3323	1.3323	77.95
CM	$-\infty$	1.3675	$+\infty$	12.17
UBM	**−0.2568**	1.3321	**1.3093**	113.53
SDCM	−0.2974	1.3315	1.3588	15.53
CCM	−0.2807	1.3406	1.3412	2,993.03
DBN	−0.2680	1.3311	1.3217	1,661.16
SDBN	−0.2940	**1.3270**	1.3538	17.42

paper [75]. To measure the quality of click models, we use log-likelihood and perplexity, both the simple and conditional versions, which are the most commonly used evaluation metrics for click models. In addition, we report the time it took us to train each model using a single CPU core[6] and the PyPy interpreter.[7]

5.4.2 Results and Discussion

The results of our experimental comparison are shown in Table 5.1. The table reports log-likelihood, perplexity, conditional perplexity and training time. The best values for each evaluation metric are indicated in boldface. Below we discuss the results for each metric.

Log-likelihood. The log-likelihood metric shows how well a model approximates the observed data, which, in our case, is clicks. Table 5.1 and Figure 5.1 report the log-likelihood values for all basic click models. Notice that the *cascade model* (CM) cannot handle query sessions with more than one click and gives zero probabilities to all clicks below the first one. For such sessions, the log-likelihood of CM is $-\infty$. Similarly, the value of conditional perplexity for CM is $+\infty$. We could consider only sessions with one click to evaluate the CM model. However, this would make the CM log-likelihood values incomparable to those of other click models.

Since most click models have the same set of attractiveness parameters that depend on queries and documents (apart from RCM and DCTR), the main difference between the models is the way they treat examination parameters (e.g., the number of examination parameters and the estimation technique).[8] Thus, it is natural to expect that models with

[6] Intel Xeon CPU E5-2650 0 @ 2.00 GHz, 20 MB L2 cache.

[7] `http://pypy.org`.

[8] Notice that DBN and SDBN differ from other models also by considering satisfaction parameters that depend on queries and documents.

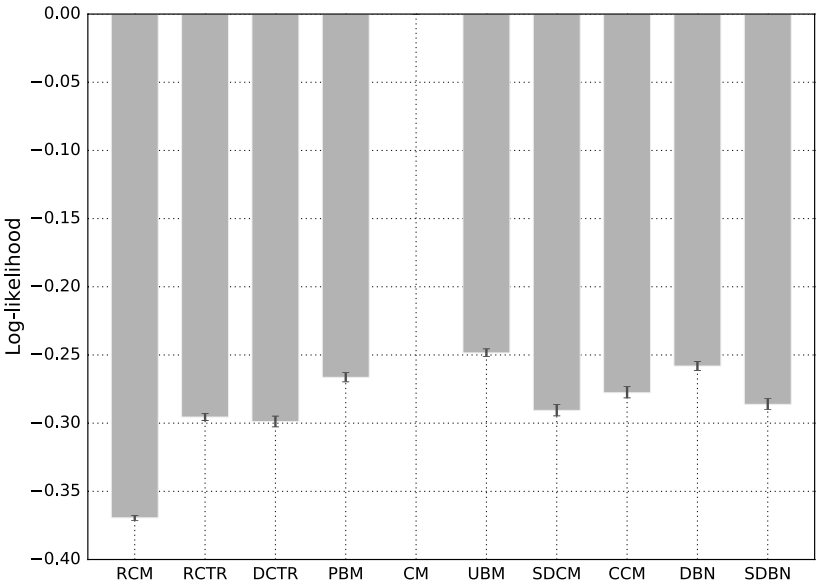


Figure 5.1: Log-likelihood values for different models; higher is better. Error bars correspond to the 95% bootstrap confidence intervals. The *cascade model* (CM) is excluded because it cannot handle multiple clicks in a session.

more examination parameters, which are estimated in connection to other parameters (i.e., using the EM algorithm), should better approximate the observed data.

The results in Table 5.1 and Figure 5.1 generally confirm this intuition. UBM, having the largest number of examination parameters, which are estimated using EM, appears to be the best model in terms of approximating user clicks based on previous clicks and skips. It is followed by DBN (ten examination parameters, a set of satisfaction parameters, EM algorithm) and PBM (ten examination parameters, EM algorithm). Other models have noticeably lower log-likelihood with the RCM baseline being significantly worse than the others as expected.

Notice that DBN outperforms PBM and SDBN outperforms SDCM (although not significantly), respectively, having fewer examination parameters and using the same estimation technique. This is due to the fact that DBN and SDBN have a large set of satisfaction parameters that also affect examination. Overall, the above results show that complex models are able to describe user clicks better than the simple CTR models, which confirms the usefulness of these models.

As expected, simple models, namely RCTR, DCTR and RCM, have the lowest log-likelihood. Notice, however, that RCTR and DCTR do not differ much from SDCM and SDBN, while being much simpler and sometimes much faster (RCTR). Still, the downside of these simple CTR-based models is that they do not explain user behavior on a SERP, as opposed to SDCM and SDBN that explicitly define examination and click behavior and that can, therefore, be used in a number of applications of click models.

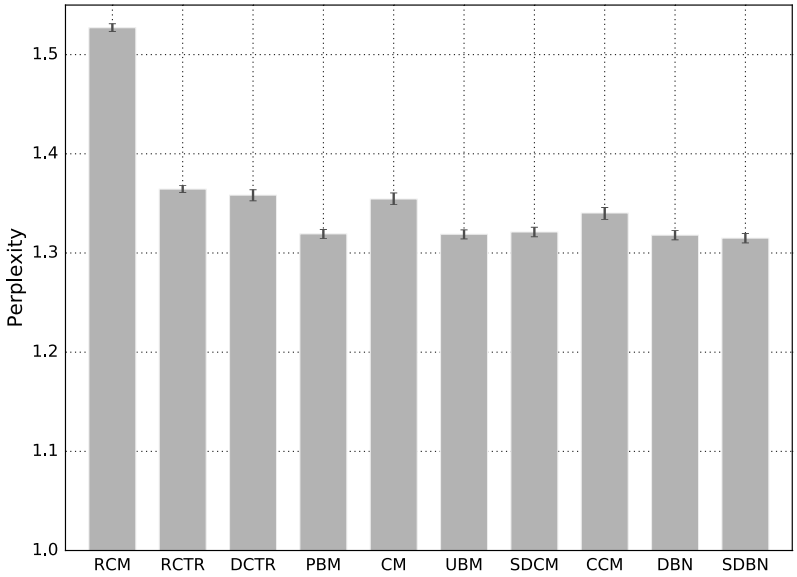


Figure 5.2: Perplexity values for different models; lower is better. Error bars correspond to the 95% bootstrap confidence intervals.

Perplexity. Perplexity shows how well a click model can predict user clicks in a query session when previous clicks in that session are not known (the lower the better). Table 5.1 shows that this version of perplexity (defined in Section 5.2.2) does not directly correlate with log-likelihood. Therefore, we believe that the simple perplexity should be preferred over the conditional perplexity for the task of click model evaluation because it gives a different perspective compared to log-likelihood.

When ranking click models based on their perplexity values (see Table 5.1 and Figures 5.2 and 5.3) the best model is SDBN followed by DBN, SDCM, UBM and PBM, respectively. The results show that complex click models perform better than CTR not only in terms of log-likelihood but also in terms of perplexity.

By looking at perplexity values for different ranks, as shown in Figure 5.3, one can see that, apart from the simple CTR-based models and the cascade model, all the models show similar perplexity results with the biggest difference observed for rank 1.

Conditional perplexity. As discussed in Section 5.2.3, conditional perplexity can be seen as a per rank decomposition of the likelihood. Indeed, Table 5.1 and Figure 5.4 show that the conditional perplexity produces the same ranking of click models as log-likelihood (apart from RCTR and DCTR, whose ranks are swapped). This means that conditional perplexity does not add much additional information when used together with log-likelihood for click model evaluation.

By looking at a per-rank decomposition (Figure 5.5) one can see a more non-trivial pattern. First, UBM is a clear leader starting from rank 4, suggesting that knowing the

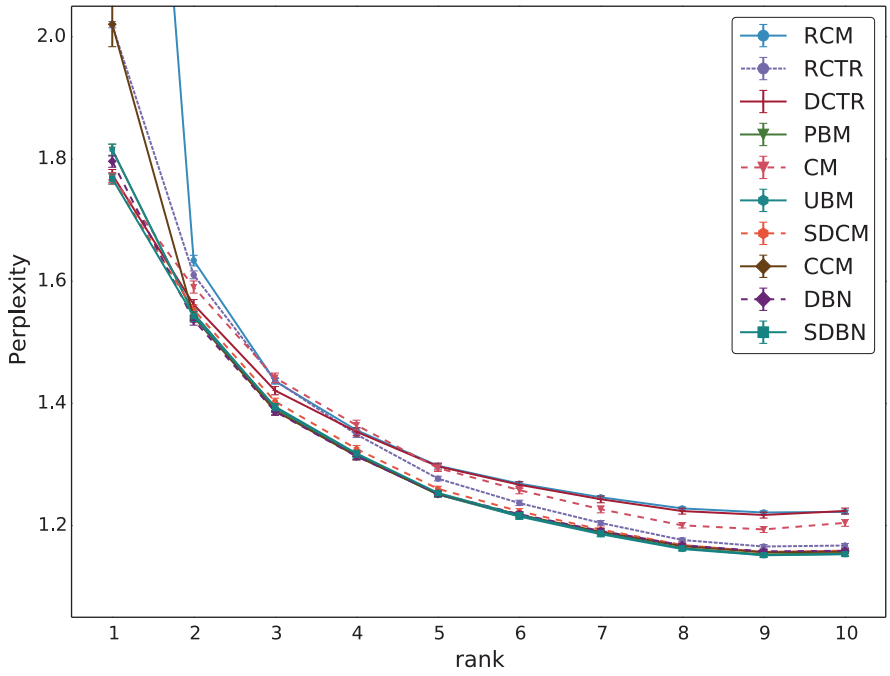


Figure 5.3: Perplexity values for different models and at different ranks; lower is better. Error bars correspond to the 95% bootstrap confidence intervals.

distance to the last click is quite useful for the examination prediction (cf. Section 3.1.5). DBN also shows quite good results for lower ranks, much better than SDBN, although not as good as UBM. Surprisingly, predicting a click on the first position is easier for SDBN than for the more complex DBN model. This suggests that the DBN assumptions work either for higher ranks or for lower ranks (with different γ parameter values), but cannot quite explain user behavior as a whole.

Training time. The amount of time required to train a click model (Table 5.1 and Figure 5.6) directly relates to the estimation method used. MLE iterates through training sessions only once and, thus, it is much faster than EM, which needs many iterations. Click models estimated using MLE require from 2 to 18 seconds to train on 750,000 query sessions, while the fastest models are RCM and RCTR. The EM-based models, instead, require from 1 to 50 minutes to be trained on the same number of sessions, whereas the slowest model is CCM. In fact, CCM is over 60% slower than DBN, which, in turn, is over ten times slower than any other model studied in this section.

One of the things to mention here is that the choice of PyPy as a Python interpreter was crucial. Just this simple switch reduced the training time of DBN from 39 hours to 28 minutes; similar dramatic drops were observed for all other models.

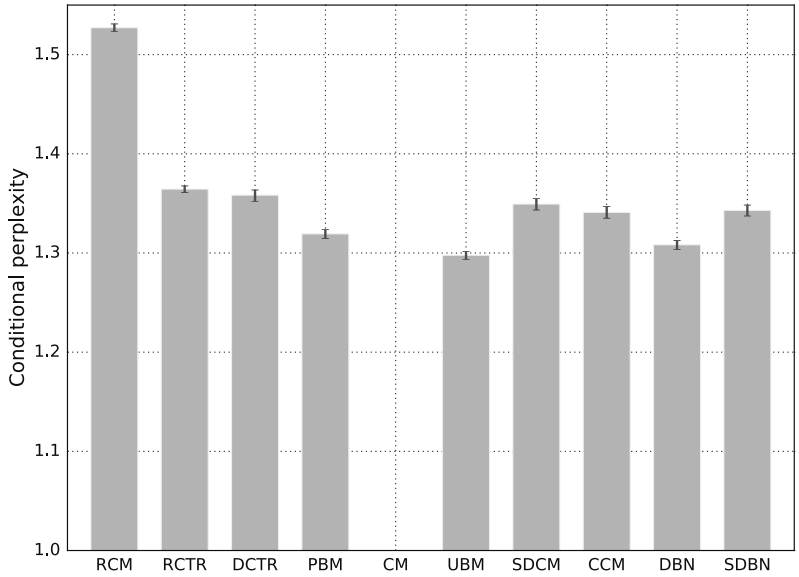


Figure 5.4: Conditional perplexity values for different models; lower is better. Error bars correspond to the 95% bootstrap confidence intervals. The *cascade model* (CM) is excluded because it cannot handle multiple clicks in a session.

5.5 Intuitiveness Evaluation: Experimental Comparison

We report here on the intuitiveness scores computed for a variety of click models, using Algorithm 5.1. For each click model we test intuitiveness with respect to the four AS factors individually, as well as the ability to capture a combination of multiple AS factors, i.e., how often a given click model's CTR agrees with multiple AS component metrics at the same time. The models that we test are: mFCM, mFCM-NO, FCM, RBP, RCM.

RBP and RCM are described in Section 3.1. The probability of a click given examination in RBP is approximated using relevance labels:[9] $P(C_r = 1 \mid E_r = 1) = R_r$; a similar model was used as a baseline model in [25].

Other models are described below.

5.5.1 Vertical-aware Click Models

Chen et al. [28] found that about 15% of the search result pages contain more than one type of vertical. Since this is a significant fraction of the search traffic, we want to adequately evaluate click models that capture user behavior in such multi-vertical settings. In order to demonstrate how our method rates such models, we introduce a *multi-vertical federated click model* (mFCM), a generalization of the *federated click model* (FCM) by

[9] For simplicity we use binary relevance labels, following [40].

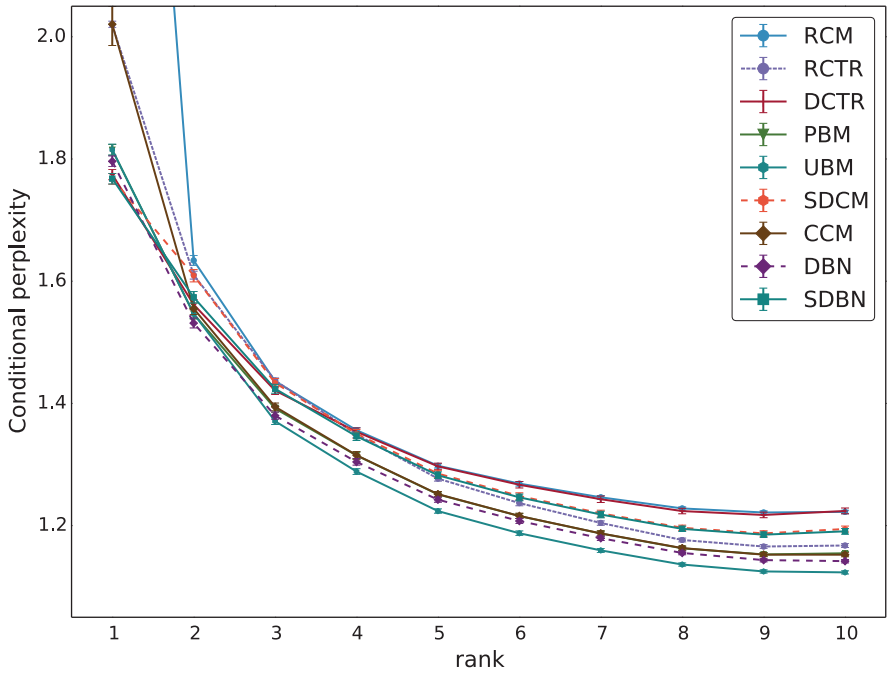


Figure 5.5: Conditional perplexity values for different models and at different ranks; lower is better. Error bars correspond to the 95% bootstrap confidence intervals. The cascade model (CM) is excluded because it cannot handle multiple clicks in a session.

Chen et al. [28, 40] in which we allow different vertical types, each with its own influence on examination probabilities.

As in [28], $P(E_r = 1)$ is influenced by the distance to vertical blocks on the SERP and the *attention bias* caused by these verticals. If there is no attention bias present, the examination probability ϕ_r depends only on the rank of the document r:

$$P\left(E_r = 1 \mid \mathbf{A}\right) = \phi_r + (1 - \phi_r) \cdot \beta_r(\mathbf{A}) \tag{5.9}$$

$$P(C_r = 1 \mid E_r = 0) = 0 \tag{5.10}$$

$$P(C_r = 1 \mid E_r = 1) = R_r. \tag{5.11}$$

Here, $\mathbf{A}$ is the vector of independent binary random variables A_j, attention bias values for each vertical $vert_j$. The influence of vertical documents on the examination probability of a document r is represented by a function $\beta_r(\mathbf{A})$. We set it to 1 if document r is a vertical document itself and decrease it as document u_r is further away from the vertical documents [28]. According to Chen et al. [28], the decrease should depend on the vertical type j, so we introduce parameters γ_j that depend solely on the vertical type j:

$$\beta_r(\mathbf{A}) = \min\left(1, \max_{j:A_j=1} \frac{1}{|dist_j(r)| + \gamma_j}\right), \tag{5.12}$$

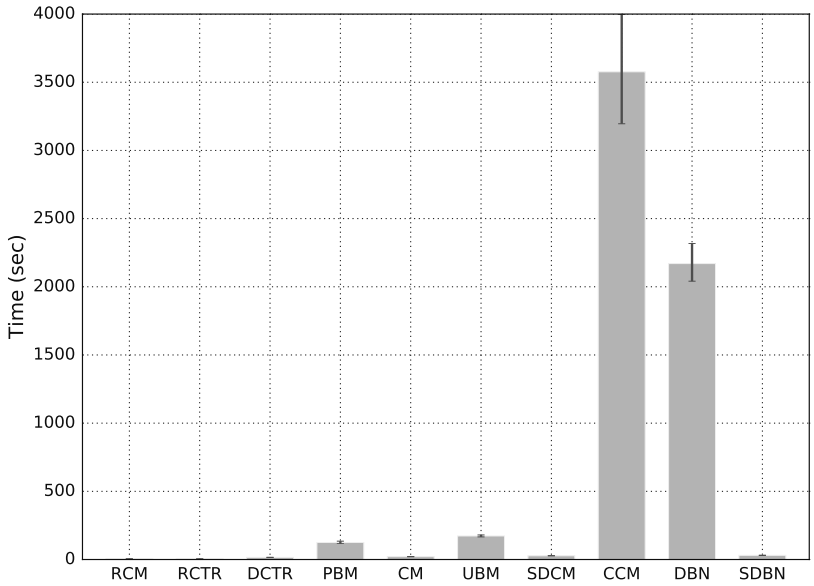


Figure 5.6: Training time for different click models; lower is better.

where $dist_j(r)$ is the distance from document u_r to the nearest document that belongs to $vert_j$ [28, 40].

If we do not distinguish between different verticals in (5.12), i.e., set $\gamma_j = \gamma$ for all j, and also assume that for a vertical j, its attention bias A_j is determined only by its position on the page:

$$P(A_j = 1) = hpos_{vert_j}, \tag{5.13}$$

we obtain the FCM model exactly as it was used in [40].

In order to distinguish verticals, we use the *vertical orientation*—the probability that users prefer a certain vertical to general web results [171, 174]. We write $orient(vert_j, q)$ to denote the orientation towards the type of $vert_j$, given query q. Having orientation values, we can further improve our click model by refining the estimation of attention bias:

$$P(A_j = 1) = orient(vert_j, q) \cdot hpos_{vert_j}. \tag{5.14}$$

The model defined by equations (5.9)–(5.12), (5.14) is called mFCM. The simpler *multi-vertical federated click model (no orientation)* (mFCM-NO) model that does not use vertical orientation[10] (5.9)–(5.13) is also of interest, since vertical orientation values are not always available and it is important to understand their contribution.

[10]The prefix "-NO" in the model name stands for "no orientation."

5.5.2 Data and Parameter Settings

Following Zhang et al. [167], we set $p = 0.73$ in the position-based model. For the mFCM model we instantiate the γ, ϕ and $hpos$ parameters similar to [40]. We set γ to 0.1 for multimedia verticals such as News or Blogs and 0.2 for text-based verticals such as Image or Video to resemble click heatmaps reported by [28] for the corresponding vertical types. We also set $hpos = [.95, .9, .85, .8, .75, .7, .3, .25, .2, .15]$ for multimedia verticals as in [40] (assuming the user cannot see documents below rank 6 without scrolling), and for text verticals $hpos = [.95, .3, .25, .15, .10, .05, .05, .05, .05, .05]$, since Chen et al. [28] suggest that a text vertical, unlike multimedia verticals, does not substantially influence user clicks if it is not at the top of the page; this is also supported by [154]. As in [40], ϕ equals $[.68, .61, .48, .34, .28, .2, .11, .1, .08, .06]$ based on the eye fixation probabilities reported by Joachims et al. [92].

To complete the experimental setup we need to specify the aggregated search systems and document dataset that we use. We use simulated aggregated systems from [174], which are built by systematically varying the quality of key aggregated search components. Specifically, we use four state-of-the-art VS systems. We also employ three ranking functions (BM25, TF and Perfect) for selecting vertical items for IS and three ways (Perfect, Random and Reverse) to embed vertical result blocks on the final aggregated search pages (RP). In total, we simulate 36 aggregated search systems ($4 \times 3 \times 3$). As a document collection we use a public aggregated search dataset from the TREC 2013 FedWeb track [120] for which relevance judgements of documents and vertical orientation preference judgments are available for each topic (query). There are 50 topics in our collection, so with 36 simulated aggregated search systems runs we have a total of $C^2_{36} = 630$ run pairs and $50 \cdot 630 = 31{,}500$ pairs of result pages.

5.5.3 Results

Table 5.2 lists our results. For every gold standard metric and every pair of click models, we give the intuitiveness scores of both models in the pair and the percentage of result page pairs for which the models disagree.

For example, Table 5.2(a) shows that if we compare mFCM and mFCM-NO in terms of the component VS (the ability to select relevant verticals), there are 14.7% ($4{,}620$ out of $31{,}500$ pairs) disagreements. The intuitiveness score for mFCM is 0.870, which is the fraction of these disagreements for which mFCM agrees with the gold standard metric. The score for mFCM-NO lies at 0.833, so mFCM is more likely to agree with the VS metric than mFCM-NO. Note that the scores of two competing models do not add up to 1; when the gold standard judges two result pages to be equally good, both click models agree with the gold standard. That is also why the scores for a very simple *random click model* (RCM) are relatively high in Table 5.2(a). We can also observe that as two click models differ more, the percentage of disagreements increases. For instance, the more complex click models tend to have a substantial disagreement with the RCM.

Let "$CM_1 > CM_2$" denote the relationship "click model CM_1 statistically significantly outperforms click model CM_2 in terms of concordance with a given gold-standard metric."

Our findings can be summarized as follows:

- VS: mFCM > mFCM-NO, FCM;
- VD: mFCM, mFCM-NO > PBM > FCM > RCM;
- IS: mFCM > mFCM-NO > FCM > PBM > RCM;
- RP: FCM > RBP > mFCM, mFCM-NO > RCM;
- VS and IS: mFCM > mFCM-NO > FCM > RBP > RCM;
- VS, IS, VD: mFCM > mFCM-NO > FCM > RBP > RCM;
- VS, IS, RP, VD: mFCM > RBP > RCM, FCM > mFCM-NO > RBP > RCM.

For single-component evaluation, mFCM outperforms the other models on VS, VD and IS, with mFCM-NO as a second-best alternative. The same holds for the combinations VS + IS and VS + IS + VD. For RP, FCM performs best, with RBP ranking second. The RP factor is measured as correlation with a "perfect" result page in which verticals with high vertical orientation are put on top. However, this order does not necessarily emit the maximum number of clicks in FCM-like click models. For example, if there is a vertical lower on the page that attracts a lot of attention, it may be better to place the relevant document just above or below this vertical. The table suggests that the intrinsic "optimal" result orders for mFCM and mFCM-NO are further from the "perfect" order than RBP's. When we look at all gold standard metrics combined, FCM and mFCM are almost equally good, with mFCM-NO again as a second-best alternative.

Our evaluation method implies that mFCM captures the VS, IS and VD factors very well, better than any other model we tested. Even without orientation values, mFCM-NO is able to capture these factors. mFCM performs worse at capturing result presentation as measured by our RP metric, which is unsurprising as the multiple vertical click model focuses less on putting relevant results on top and better accounts for attention bias caused by multiple vertical blocks. This shows that our intuitiveness evaluation method is able to draw non-trivial detailed conclusions about model's performance.

5.6 Conclusion

In this section we look back at the research questions formulated at the beginning of the chapter and show how we answered them. The questions are as follows:

RQ2.1 How do different click models perform when evaluated on a common dataset?

RQ2.2 How should we evaluate click models for complex aggregated SERPs?

We first presented a detailed introduction into common evaluation metrics—metrics that can be used for both simple and vertical-aware click models. We performed a detailed analysis of all basic click models on one common open dataset, with models implemented in an open-source software package, thus addressing **RQ2.1**. We found that more complex click models, namely PBM, UBM, SDCM, CCM, DBN and SDBN, outperform CTR-based models and the cascade model both in terms of log-likelihood and perplexity. Also,

the models whose parameters are estimated using the EM algorithm (PBM, UBM, CCM and DBN) tend to outperform those estimated using the MLE technique (SDCM and SDBN), because in the former case model parameters are estimated in connection to each other. However, such models take much longer to train. When comparing pairs of more complex and simplified models (DBN/SDBN and CCM/SDCM) one can see that the simplified one yields better perplexity values but worse conditional perplexity and likelihood scores. We believe that these simplifications together with the MLE algorithm can find more robust estimations for the model parameters, but cannot fully interpret the relation between past and future behavior in the same session.

There is no clear winner among the best performing models, but UBM tends to have the highest log-likelihood and almost the lowest perplexity. A potential drawback of UBM, however, is that the interpretation of its examination parameters is not straightforward (see Section 3.1.5). UBM is closely followed by DBN and PBM, which sometimes have slightly worse performance, but their parameters are easier to interpret.

We then turned our attention to vertical-aware models to answer **RQ2.2**. We introduced an evaluation method that can be used to assess a vertical-aware click model's ability to capture key components of an aggregated search system and demonstrated it using different vertical-aware as well as traditional click models. We also showed that click models that account for multiple vertical blocks within a single result page typically get higher intuitiveness scores, which indicates that our evaluation method measures the right thing. In addition, we showed that a model that uses vertical orientation values have higher intuitiveness scores than the corresponding model that does not (cf. **RQ1.2**).

Future directions. When doing basic click model evaluation we did not include click models for vertical search because there is no dataset that would allow us to do such a comparison. We also omitted some evaluation facets such as prediction of clicks vs. prediction of their absence, query frequency and query entropy analysis, to name a few. For those additional dimensions we refer to a complementary study by Grotov et al. [73].

One limitation of the intuitiveness analysis is that we did not use raw click data to infer the parameters of the click models we experiment with. However, we set these parameters using previous work that did use real click and eye gaze data [28, 92, 154].

As a direction for future work we want to compare the findings of the intuitiveness test with conventional model performance tests (e.g., perplexity of click prediction) and see whether good intuitiveness scores also imply good click prediction results and vice-versa.

The next two chapters rely heavily on the framework of click models. In Chapter 6 we introduce an important application of click models—model-based offline evaluation metrics. We further develop this approach in Chapter 7 to include signals beyond clicks and also tune the metric to fit the satisfaction reported by the users. Alternatively, the reader may proceed directly to Chapter 8 where we study online evaluation methods and use click models to simulate users in some of the experiments.

Table 5.2: Intuitiveness test results. For each pair of click models, the higher score is shown in bold, with the fraction of disagreements in parentheses. Results (a)–(d) show click model performance w.r.t. individual AS components; results (e)–(g) concern a click model's ability to capture multiple components. Significant differences (sign test) are indicated with △ ($\alpha = 0.05$) and ▲ ($\alpha = 0.01$).

Evaluation Criteria		mFCM-NO	FCM	RBP	*RCM*
(a) **(VS)** gold standard: vertical selection precision	mFCM	**0.870**/0.833△ (14.7%)	**0.865**/0.842△ (18.2%)	**0.873**/0.856 (21.5%)	0.851/**0.865** (44.5%)
	mFCM-NO	-	0.843/**0.850** (17.0%)	0.868/**0.877** (23.1%)	0.845/**0.870** (44.3%)
	FCM	-	-	0.879/**0.885** (22.0%)	0.849/**0.871** (44.2%)
	RBP	-	-	-	0.848/**0.868** (45.3%)
(b) **(VD)** gold standard: vertical recall	mFCM	**0.819**/0.812 (14.7%)	**0.879**/0.713▲ (18.2%)	**0.832**/0.748▲ (21.5%)	**0.860**/0.678▲ (44.5%)
	mFCM-NO	-	**0.884**/0.715▲ (17.0%)	**0.817**/0.743▲ (23.1%)	**0.858**/0.677▲ (44.3%)
	FCM	-	-	0.763/**0.819**▽ (22.0%)	**0.822**/0.708▲ (44.2%)
	RBP	-	-	-	**0.828**/0.688▲ (45.3%)

(c) **(IS)** gold standard: mean precision of vertical retrieved items	mFCM	**0.765**/0.732△ (14.7%)	**0.754**/0.691▲ (18.2%)	**0.832**/0.515▲ (21.5%)	**0.918**/0.349▲ (44.5%)
	mFCM-NO	-	**0.745**/0.706△ (17.0%)	**0.815**/0.542▲ (23.1%)	**0.912**/0.352▲ (44.3%)
	FCM	-	-	**0.805**/0.549▲ (22.0%)	**0.902**/0.357▲ (44.2%)
	RBP	-	-	-	**0.828**/0.423▲ (45.3%)
(d). **(RP)** gold standard: Spearman Correlation with "perfect" aggregated search page	mFCM	**0.601**/0.592 (14.7%)	0.493/**0.691**▼ (18.2%)	0.575/**0.649**▼ (21.5%)	**0.643**/0.551▲ (44.5%)
	mFCM-NO	-	0.477/**0.702**▼ (17.0%)	0.569/**0.644**▼ (23.1%)	**0.642**/0.553▲ (44.3%)
	FCM	-	-	**0.653**/0.576▲ (22.0%)	**0.683**/0.513▲ (44.2%)
	RBP	-	-	-	**0.650**/0.527▲ (45.3%)
(e). **(VS + IS)** gold standard: vertical selection precision AND vertical item mean precision	mFCM	**0.666**/0.606△ (14.7%)	**0.651**/0.584▲ (18.2%)	**0.728**/0.432▲ (21.5%)	**0.782**/0.294▲ (44.5%)
	mFCM-NO	-	**0.630**/0.608△ (17.0%)	**0.709**/0.474▲ (23.1%)	**0.772**/0.301▲ (44.3%)
	FCM	-	-	**0.714**/0.486▲ (22.0%)	**0.766**/0.303▲ (44.2%)
	RBP	-	-	-	**0.700**/0.362▲

					(45.3%)
(f). **(VS + IS + VD)** gold standard: vertical selection precision AND vertical item mean precision AND vertical recall	mFCM	**0.567**/0.522△ (14.7%)	**0.585**/0.456▲ (18.2%)	**0.605**/0.347▲ (21.5%)	**0.680**/0.248▲ (44.5%)
	mFCM-NO	-	**0.569**/0.469▲ (17.0%)	**0.580**/0.372▲ (23.1%)	**0.673**/0.253▲ (44.3%)
	FCM	-	-	**0.568**/0.430▲ (22.0%)	**0.649**/0.269▲ (44.2%)
	RBP	-	-	-	**0.601**/0.300▲ (45.3%)
(g). **(VS + IS + RP + VD)** gold standard: ALL single-component metrics	mFCM	0.372/**0.373** (14.7%)	0.346/**0.366** (18.2%)	**0.394**/0.257▲ (21.5%)	**0.485**/0.164▲ (44.5%)
	mFCM-NO	-	0.350/**0.370**▽ (17.0%)	**0.390**/0.263▲ (23.1%)	**0.488**/0.164▲ (44.3%)
	FCM	-	-	**0.414**/0.269▲ (22.0%)	**0.486**/0.155▲ (44.2%)
	RBP	-	-	-	**0.435**/0.187▲ (45.3%)

6

Offline Evaluation Based on Click Models

Many models have been proposed recently that are aimed at predicting clicks of web search users. In addition, some information retrieval evaluation metrics have been built on top of user models. In this chapter we bring these two directions together and propose a common approach to converting any click model into an evaluation metric. We then put the resulting *model-based* metrics as well as traditional metrics such as DCG or Precision (see Section 3.2) into a common evaluation framework and compare them along a number of dimensions.

One of the dimensions we are particularly interested in is the agreement between offline and online experimental outcomes. It is widely believed, especially in an industrial setting, that online A/B-testing and interleaving experiments (Section 2.5) are generally better at capturing system quality than offline measurements [27, 99]. We show that offline metrics that are based on click models are more strongly correlated with online experimental outcomes than traditional offline metrics, especially in situations when we have incomplete relevance judgements.

The main research questions that we address in this chapter are:

RQ3.1 Can we make use of click models to build better evaluation metrics? How do such click model-based IR metrics differ from traditional offline metrics?

RQ3.2 Which evaluation metrics are better tied to the user? Do click model-based metrics show higher agreement with online experiments? How do they compare in terms of discriminative power?

RQ3.3 How well do different offline metrics perform in the presence of unjudged documents?

RQ3.4 How can we modify offline metrics to enhance agreement with online experiments?

Our main contributions in this chapter are a method for converting click models into click model-based offline metrics. Secondly, we present a thorough analysis and comparison of specific click model-based metrics with online measurements and traditional offline metrics.

6.1 Introduction

There are currently two orthogonal approaches to evaluating the quality of ranking systems. The first approach is usually called the Cranfield approach [57] and is done *offline*. It uses a fixed set of queries and documents judged by trained people (assessors). Ranking systems are then evaluated by comparing how good their ranked lists are. Among other things, a system is expected to place relevant documents higher than irrelevant ones.

Another approach makes use of real *online* users by assigning some portion of the users to test groups (also called *flights*). The simplest variant, called *A/B-testing*, randomly assigns some users to the "control" group (these users are presented with the existing ranking results) and the "treatment" group (these users are presented with the results of an experimental ranking system). Ranking systems are then compared by analysing the clicks of the users in the "control" group against those in the "treatment" group. In the *interleaving* method by Joachims [90] users are presented with a combined list made out of two rankings. Then the system that receives more clicks is assumed to be better.

One of the main advantages of online evaluation schemes is that they are user-based and, as a result, often assumed to give us more realistic insights into the real system quality. Interleaving experiments are now widely being used by large commercial search engines like Bing and Yahoo! [27, 123] as well as studied in academia [80, 127]. However, they are harder to reproduce than offline measurements, whereas in the traditional Cranfield approach one can re-use the same set of judged documents to evaluate any ranking. This makes the use of offline rater-based evaluation methods unavoidable during the early development phase of ranking algorithms. One should take care, however, that the resulting rater-based measurements agree with the outcomes of online experiments—online comparison is often used as the final validation step before releasing a new version of a ranking algorithm.

In order to bring the two evaluation approaches closer to each other, we propose a method for building an IR metric from a *click model*. We previously discussed some basic click models in Section 3.1 and then studied them in Chapter 5. We hypothesize that click models can be turned into offline metrics and the resulting *click model-based metrics* should be closely tied to the user and hence should better correlate with online measurements than traditional offline metrics. In addition, there is a growing trend to ground offline metrics in a user model and that is exactly what click modeling does—trying to propose a better user model. So, the question is why not use better user models, based on click behavior, as the basis for offline metrics?

We put our proposal for transforming click models into metrics to the test through a set of thorough comparisons with online measurements. Our comparison includes an analysis of correlations with the outcomes of interleaving experiments, an analysis of correlations with absolute online metrics, an analysis of correlations between traditional offline metrics and our new click model-based metrics, as well as an analysis of the discriminative power of the various metrics. One dimension to which we devote special attention in our comparison framework concerns unjudged documents. As was shown by Buckley and Voorhees [18], having partially-judged result pages in the evaluation pool may result in biased measurements. We examine how different offline metrics handle this problem. We also show that in situations where we cannot afford to use only fully-judged data, we can still make good use of the available data by making adjustments, either by a

technique called *condensation* [131] or by a new *thresholding* method that we propose.

The rest of the chapter is organized as follows. Section 6.2 presents related work. Section 6.3 shows how to transform a click model into a model-based offline metric. In Section 6.4 we examine click model-based and traditional offline metrics and report on their performance. We finish with a conclusion and discussion in Section 6.5.

6.2 Related Work

Determining and comparing the quality of information retrieval systems has always been an important task in IR, both in academic and industrial research. In recent years, competition between large commercial search systems has reached the point where even a small improvement can be of great importance. As a result, a broad range of metrics to assess system performance have been proposed: *discounted cumulative gain* (DCG) by Järvelin and Kekäläinen [89], *expected reciprocal rank* (ERR) by Chapelle et al. [26], *expected browsing utility* (EBU) by Yilmaz et al. [164], to name just a few. They were also assessed from a variety of angles (see, e.g., [21, 26, 130]).

Some IR metrics have an underlying user model (e.g., ERR and EBU do) or they can be viewed as such (see [21]). However, there is still a big gap between user models and metrics. For example, some of the widely used click models, such as *user browsing model* (UBM) by Dupret and Piwowarski [69] (Section 3.1.5) and *dependent click model* (DCM) by Guo et al. [75] (Section 3.1.7), have so far not been used to develop an offline metric. Moreover, since the introduction of these early click models, many more click models have been developed, not only as improvements to previous models [74, 106], but also to address specific modeling issues, such as click models for vertical search [28, 41], models that use mouse movements along with clicks [88], or to sessions-level click models [169]. We believe that all these models can be converted into evaluation measures.

Another important group of closely related studies concerns user-based online evaluation that we discuss in Section 2.5. Apart from that, we must mention some work that touches on the comparison of offline and online evaluation. Radlinski and Craswell [123] analyzed and compared the sensitivity of both *interleaving* and traditional *offline* IR metrics against each other. They found that the outcomes of interleaving experiments generally agree quite well with offline metrics while data can be collected at a much lower cost. Below, we apply the same type of analysis to evaluate click model-based metrics and to compare them against traditional IR metrics. Ali and Chang [2] showed that *per-query* correlation between *offline* side-by-side comparisons and online *interleaving* experiments is low even when query filtering is applied. This finding suggests that aggregating results from multiple queries as was done in [123] is less noisy than computing correlations on a per query basis. Yue et al. [166] proposed ways to increase the signal of an interleaving experiment; inspired by this idea we propose to tune offline metrics through two techniques referred to as *condensation* and *thresholding* below to enhance the agreement with interleaving (see Section 6.4.1).

6.3 Click Model-based Metrics

From an initial focus on Precision as a metric, the area of web search evaluation has evolved considerably. An early lesson was that we need to apply some sort of discount to the documents that appear lower in the ranking. One of the first metrics to operationalize this idea was *discounted cumulative gain* (DCG) [89]. This metric is still widely used in the IR community. However, it has some drawbacks. One is that its discount function is not motivated by a user model. Another important issue with this metric is that it is a *static* metric, i.e., its discount values are fixed numbers. As was shown by Yilmaz et al. [164], a *dynamic* metric—a metric that dynamically assigns different discount values according to the relevance of the documents appearing higher in the ranking—more accurately represents real user behavior.

In this chapter we introduce the notion of *click model-based* metrics. The main constituent of such a model-based metric is a *click model*—a probabilistic model aimed at predicting user clicks. Apart from click events (C_r), a click model usually has hidden variables corresponding to events such as "the user examined the snippet at rank r" (E_r). All click models that we study in this chapter assume that users click a document only after examining the document's snippet, i.e., $P(C_r = 1 \mid E_r = 0) = 0$ (cf. (3.2)).

Following Carterette [21], we distinguish between *utility-based* metrics and *effort-based* metrics. These give rise to two ways of mapping a click model to a click model-based offline metric. First, a utility-based metric uses a click model only to predict the click probability $P(C_r = 1)$ for the r-th snippet in the ranking. This probability is then used to calculate the metric value as the *expected utility*:

$$uMetric = \sum_{r=1}^{n} P(C_r = 1) \cdot R_r, \tag{6.1}$$

where R_r is the relevance of the r-th document. It is common to use four or five relevance grades that are further mapped to numeric values. For example, the TREC 2011 Web Track [54] uses four levels of relevance: from 0 for *Irrelevant* documents to 3 for *Highly Relevant* documents.

Second, an effort-based metric requires a click model to have a notion of "user satisfaction" (S_r). A click model must have hidden variables S_r such that $P(S_r = 1 \mid C_r = 0) = 0$ (the user can only be satisfied by the documents she clicked) and $P(E_t = 1 \mid S_r = 1) = 0$ for $t > r$ (after being satisfied the user stops examining documents). Having this, we can define a metric to be an expected value of some *effort function*[1] at the stopping position:

$$rrMetric = \sum_{r=1}^{n} P(S_r = 1) \cdot \frac{1}{r} = \sum_{r=1}^{n} \sigma_r P(C_r = 1) \cdot \frac{1}{r}, \tag{6.2}$$

where $\sigma_r = P(S_r = 1 \mid C_r = 1)$ is a *satisfaction probability*.

A click model is usually *trained* using a click log. As a result we get values of the model parameters that can further be used to calculate the probability of clicks or satisfaction events to use in (6.1) and (6.2). Some parameters are just constants, some

[1] Following [21] we use reciprocal rank $\frac{1}{r}$ as an effort function. While we are not doing it here, it would be interesting to evaluate metrics with different effort functions.

Table 6.1: Click model-based metrics and their underlying models. Previously proposed models/metrics are followed by the reference.

	Derived metric	
Underlying click model	Utility-based	Effort-based
CM [59] / DBN [25]	uSDBN [26]	ERR [26]
DBN [25]	EBU [164]	rrDBN
DCM [75]	uDCM	rrDCM
UBM [69]	uUBM	–

depend on the position(s) in the ranking and some depend on the document and/or query. Parameters of the last type are the hardest ones to be used in a metric, as we want our metric to work even for previously unseen documents. But, fortunately, parameters of this type can usually be approximated from the document's relevance. In fact, when training a model we assume that these parameters only depend on the document relevance and not on the document itself. We will demonstrate this procedure for the *attractiveness* parameters in DBN, DCM, UBM and for the *satisfaction* parameters in DBN.

If a model meets the requirements listed above, it can be transformed into a click model-based metric. There is no step-by-step algorithm for such a transformation, only general guidelines. In the following sections we demonstrate the idea, using well-known click models as an example. We want to stress, however, that our framework is general enough to be applied to other click models, including those that use additional sources of information, such as recently studied session-based click models [169] or click models for vertical search [28, 41].

In Table 6.1 we classify previously studied metrics (ERR by Chapelle et al. [26], EBU by Yilmaz et al. [164]) and propose several new click model-based metrics: rrDBN, uDCM, rrDCM, uUBM. The left most column lists click models, the center and right most column denote derived offline metrics, utility-based and effort-based, respectively. As a recipe for naming a metric, we use the name of the underlying model and prefix it with the type metric that we are defining: *u-* for *utility-based* and *rr-* for reciprocal rank *effort-based* metrics.

6.3.1 Previously Studied Metrics

In this section we show how two previously proposed metrics, ERR and EBU, can be viewed as click model-based metrics. Despite the fact that they are different and were not in fact proposed as derivatives of a click model, they can both be viewed as metrics based on special cases of the *dynamic Bayesian network* (DBN) model (Section 3.1.6).

Expected reciprocal rank (ERR) is based on the *cascade model* (CM), which, as we mentioned in Section 3.1.6, is a simplification of DBN. If we opt for a DBN simplification where we set $\alpha_r \equiv 1$, the probability of clicking the r-th document is as follows:

$$P(C_r = 1) = P(E_r = 1) = \gamma^{r-1} \prod_{i=1}^{r-1} (1 - \rho_i), \tag{6.3}$$

where ρ_i is the probability of i-th document being relevant and γ is the continuation probability.[2] Correspondingly, the probability of satisfaction

$$P(S_r = 1) = \rho_r P(C_r = 1) = \rho_r \gamma^{r-1} \prod_{i=1}^{r-1} (1 - \rho_i). \tag{6.4}$$

The probability of being relevant is usually viewed as a mapping $R \to \rho$ from the relevance grades to the interval $[0, 1]$. In the original ERR paper [26] the authors use the mapping (3.37) motivated by DCG, but one may also fit this mapping from a click log.

Using probabilities from (6.3) and (6.4), we end up with the ERR and uSDBN metrics (cf. (6.2), (6.1), (3.41) and [26]):

$$\text{ERR} = \sum_{r=1}^{n} \left(\rho_r \gamma^{r-1} \prod_{i=1}^{r-1} (1 - \rho_i) \right) \cdot \frac{1}{r} \tag{6.5}$$

$$\text{uSDBN} = \sum_{r=1}^{n} \left(\gamma^{r-1} \prod_{i=1}^{r-1} (1 - \rho_i) \right) \cdot \rho_r. \tag{6.6}$$

In the original version of the ERR metric, the continuation probability γ of the DBN model was set to 1 and we do likewise. Conversely, for uSDBN we set the continuation probability γ to 0.9, as suggested in [25].

The *expected browsing utility* (EBU) metric by Yilmaz et al. [164] is also based on a variation of the DBN model. Unlike the original DBN model, their modification allows for different continuation probabilities in different situations ($p_{cont|click}$, $p_{cont|nonrel}$, $p_{cont|rel}$). While these parameters lead to greater flexibility in setting up the metric, they also represent a difficult choice for a practitioner to make. They were all set to 1 in the original paper [164] and here we do the same. By doing so we reduce the underlying model to DBN [25] with continuation probability $\gamma = 1$.

One notable difference between the ERR and EBU metrics is that EBU does not set the attractiveness probabilities to 1. Instead, the attractiveness probabilities and satisfaction probabilities are both estimated from a click log using the assumptions that they are determined by the document relevance:

$$\begin{aligned} \alpha_r &\approx P(C_r \mid R_r) \\ \sigma_r &\approx P(S_r \mid R_r), \end{aligned}$$

where C_r is the random variable corresponding to a click on the r-th document, S_r is the random variable corresponding to leaving the result page after clicking the r-th document (satisfaction) and R_{u_r} is the relevance of the r-th document u_r.

6.3.2 New Click Model-based Metrics

In this section we propose new offline metrics by introducing an *effort-based* variant of the EBU metric and also by converting the two popular click models, UBM and DCM, into

[2] Here we also decided to keep γ from the DBN model, unlike the original paper which set it to 1 to fully reduce DBN to CM.

click model-based metrics. By doing so we show that our framework of *click model-based* metrics is not only a way of interpreting previously studied metrics, but also a way of deriving new metrics in a principled way.

The rrDBN metric uses essentially the same user model as the EBU metric. In fact, the parameters for EBU and rrDBN are the same. The only difference is that rrDBN is calculated using (6.2) instead of (6.1).

Next, the uDCM and rrDCM metrics can be derived from the *dependent click model* (DCM) (Section 3.1.7) in a way similar to how EBU and rrDBN are derived from DBN.

As was shown by Turpin et al. [157], the attractiveness of a document's snippet can be approximated as a function of its relevance grade. A mapping from grades to attractiveness probabilities can be inferred from a click log using the click model (DCM in this case).[3] For this purpose we impose the constraint that documents with the same relevance have the same attractiveness, i.e., the attractiveness of a document is a function of its relevance grade: $\alpha_r = \alpha(R_r)$.

Finally, using the click model and equations (6.1), (6.2), we can define the uDCM and rrDCM metrics as follows:

$$\begin{aligned} uDCM &= \sum_{r=1}^{N} \alpha(R_r) \prod_{i=1}^{r-1} (1 - \alpha(R_i)\sigma_i) \cdot R_r \\ rrDCM &= \sum_{r=1}^{N} \sigma_r \alpha(R_r) \prod_{i=1}^{r-1} (1 - \alpha(R_i)\sigma_i) \cdot \frac{1}{r}. \end{aligned}$$

Chen et al. [28] report that the *user browsing model* (UBM) [69] performs better than DBN in terms of click prediction perplexity. We also evaluated this model using a Yandex click log. A sample of clicks collected in November 2012 was used. We removed pages without clicks and split the remaining data into training and test set. In total, we had 1,191,963 training and 1,292,993 test pages. To compare the models we used perplexity gain (5.6). On our data UBM outperforms DBN by 16% which is quite substantial.

This finding motivates the idea of deriving an offline metric from UBM. In the UBM model the click probability is governed by the attraction bias and the examination bias (see Section 3.1.5 for more details):

$$P(C_r = 1 \mid \mathbf{C}_{<r}) = P(A_r = 1)P(E_r = 1 \mid \mathbf{C}_{<r}) = \alpha_r \gamma_{rr'},$$

where C stands for click, A for attraction, E for examination; u is the document URL, q is the user query, r is the document rank (position), and $r' = \max\{j < r \mid C_j = 1\}$ is the rank of the previous click.[4]

Like for the EBU/rrDBN and uDCM/rrDCM metrics, we assume that the attractiveness probability α is a function of the relevance of the document: $\alpha_r = \alpha(R_r)$. The examination probabilities $\gamma_{rr'}$ can be precomputed from a click log during the model training process. One important difference from the previously studied models is that UBM relies on previous clicks and these are not available offline. To deal with this

[3] The source code for probabilistic inference is freely available at `https://github.com/varepsilon/clickmodels#train_for_metric`.

[4] As in [69] we use a virtual 0th position (which is assumed to be always clicked) to simplify our equations.

problem we factorize the probability $P(C_r = 1)$ over the position of previous clicks r':

$$P(C_r = 1) = \sum_{r'=0}^{r-1} P(C_{r'} = 1, C_{r'+1} = 0, \ldots, C_{r-1} = 0, C_r = 1).$$

By applying Bayes' rule we get

$$\begin{aligned} P(C_r = 1) = &\sum_{r'=0}^{r-1} P(C_{r'} = 1) \\ &\cdot \prod_{k=r'+1}^{r-1} P(C_k = 0 \mid C_{r'} = 1, C_{r'+1} = 0, \ldots, C_{k-1} = 0) \\ &\cdot P(C_r = 1 \mid C_{r'} = 1, C_{r'+1} = 0, \ldots, C_{r-1} = 0) = \\ = &\sum_{r'=0}^{r-1} P(C_{r'} = 1) \left(\prod_{k=r'+1}^{r-1} (1 - \alpha_k \gamma_{kr'}) \right) \alpha_r \gamma_{rr'}. \end{aligned}$$

Finally, the click probability is given by a recursive formula:

$$\begin{aligned} P(C_0 = 1) &= 1 \\ P(C_r = 1) &= \sum_{r'=0}^{r-1} P(C_{r'} = 1) \left(\prod_{k=r'+1}^{r-1} (1 - \alpha_k \gamma_{kr'}) \right) \alpha_r \gamma_{rr'}, \end{aligned}$$

where $\alpha_r = \alpha(R_r)$, and α and γ are known functions estimated from clicks. It is important to note, that unlike Dupret and Piwowarski [69], we used all queries, not only queries with high clickthrough rate. So our resulting γ function[5] is different from that analysed by Dupret and Piwowarski, and might be interesting on its own. For example, $\gamma_{rr'}$ is much less than 1 for $r' > 0$ which corresponds to the fact that many users click on only one document.

Given the click probability we can define the metric:

$$uUBM = \sum_{r=1}^{n} P(C_r = 1) \cdot R_r. \tag{6.7}$$

The UBM click model does not have a notion of user satisfaction and hence we do not introduce an "rrUBM" metric.

6.4 Analysis

In this section we analyze the click model-based metrics previously listed, both old and new, along a number of dimensions. We compare *click model-based* metrics to traditional offline metrics (Section 3.2). As traditional metrics we consider precision, with two possible binarizations of four scale judgments (Precision treats the highest three relevance grades—3, 2, and 1—as "relevant," while Precision2 only treats the highest two relevance

[5]The values are available in the source code under name `UBM_GAMMAS`.

grades 3 and 2 as "relevant") as well as DCG. We decided not to include the NDCG metric and thus to overcome potential issues with corpus-dependent NDCG normalization (see Section 3.2.5).

We start by determining correlations of various offline metrics to the outcomes of interleaving experiments in a way proposed by Radlinski and Craswell [123]. These correlations are then used to compare offline metrics to each other. The metric that shows the best correlation with interleaving outcomes is assumed to better represent real user behavior. We then move to more traditional comparison techniques, such as metric-to-metric correlations and discriminative power.

6.4.1 Correlation with Interleaving Outcomes

As was shown by Radlinski et al. [127], absolute click metrics are often unable to determine differences in IR systems. Moreover, they are always difficult to interpret and may even be misleading, because we cannot know for sure how these metrics are related to user satisfaction.

Fortunately, there is another approach, the *pairwise* or *interleaved* comparison techniques mentioned earlier [90, 127]. Following this approach, we compare two ranking systems by presenting a user with an interleaved result page, containing documents from both result lists. The winner is then determined from user clicks. We assess an offline IR metric m in terms of its agreement with the interleaving outcomes. Specifically, we use the *team-draft interleaving* (TDI) method by Radlinski et al. [127]. In this method each document in the interleaved page is assigned to exactly one of the two ranking systems ("the teams"). We then say that a system wins a comparison if the documents it contributes to the combined list receive more clicks. The system that wins most of the comparisons is assumed to be better.

For the current experiment we use a click log of the Yandex search engine collected in October–December 2012. During this period we focus on five revisions of the core ranking function (A, B, C, D, E), with each revision being compared to the previous one using TDI, that was run for 5–10 days. For each of our ten experiments we have at least 200,000 impressions as in the work by Radlinski and Craswell [123]. Some ranking function revisions influence more than one market (country), so in total we have ten pairs of algorithms to compare: $\Delta_1 AB$, $\Delta_2 AB$, $\Delta_1 BC$, $\Delta_1 CD$, $\Delta_2 CD$, $\Delta_3 CD$, $\Delta_4 CD$, $\Delta_1 DE$, $\Delta_2 DE$, $\Delta_3 DE$. For each algorithm pair we record the *interleaving signal* value, i.e., the deviation from 50% of the number of cases where the newer system was preferred. For instance, if in the experiment labeled $\Delta_i XY$ system Y was preferred over system X in 51% of all cases, we say that the *interleaving signal* for the experiment is 1%.

Once we have interleaving signals, we want to compare them to the signals given by offline IR measures, i.e., the average difference of the metric values. Unlike in the traditional Cranfield approach we use queries and documents from the query log. When computing an offline metric signal for a particular experiment $\Delta_i XY$, we extract queries issued by the users assigned to the experimental flight. For these queries we also extract the document lists that would have been produced by each of the systems X and Y if they had not been interleaved. By using click log-based queries when comparing the signal of an offline metric to the interleaving signal we eliminate the effect induced by the choice of a query set that one needs to compile for a Cranfield-style evaluation. Here it also

allows us to perform experiments with historical revisions of a ranking algorithm that is no longer running. Although this approach has some advantages for our research problem, it has some disadvantages for everyday usage. One notable drawback is that we use only part of the judgements available because not all the queries that we have judgements for were submitted by the users of the experimental flights. For each experiment and each metric we keep only the queries that have at least one document judged. Depending on the experiment we have from 178 to 5,815 queries per experiment (median number is 573). As was shown in [123], it is usually sufficient to have approximately 100 queries to identify the better system in an offline comparison.

The amount of data available to the search engine is usually much larger than a human can handle. Even more important is the fact that the web corpus is constantly changing, so we cannot maintain complete judgements even for a limited set of queries. That is why it seems natural that some documents returned by the system do not have relevance judgements. In order to analyse the tradeoff between adding noise from unjudged documents and reducing the noise by allowing more queries we introduce a parameter `#unjudged`. We discard queries for which the number of unjudged documents in the top ten is bigger than this value for either of the two systems taking part in a TDI experiment. We vary this bound and see how it influences the correlation between offline metrics and interleaving. For each offline metric m and each value of `#unjudged` from 1 to 9 we compute the weighted Pearson correlation (similar to [26]) between the metric signal and the interleaving signal. As a weight we use the number of queries participating in the calculation of the metric signal (this number is different for each experiment). The results are presented in Figure 6.1. We can see that the *effort-based* metrics rrDBN and rrDCM are better at dealing with unjudged documents and are remarkably different from their *utility-based* counterparts; we will confirm this difference in Section 6.4.3. Another interesting observation can be made about the Precision and Precision2 metrics. Their behavior differs and, moreover, Precision has a negative correlation and this is not the case for Precision2. This seems to be due to the fact that unjudged documents are treated in the same way as the lowest relevance grade 0, whereas in fact they have a higher chance to belong to one of the top relevance grades: 92% of the documents in the top 10 have a relevance grade higher than 0, while only 23% have a relevance grade higher than 1.

As we can see from Figure 6.1, when we increase `#unjudged`—the maximum number of unjudged documents—to 4 or higher, the correlation drops for all the metrics studied. This means that adding queries with highly incomplete judgements adds noise to the metric signals. The problem of unjudged documents has previously been studied by Sakai [131], and his proposed solution is to exclude unjudged documents from the ranked list and *condense* the remaining documents. Despite its heuristic nature, this idea actually leads to an increase in correlation for most of the metrics as shown in Figure 6.2. The exceptions from this rule are rrDBN and rrDCM that supposedly suffer most from the incorrect effort function values. For example, if we miss a judgement for the first document, then for the second document we apply a $\frac{1}{1}$ discount instead of $\frac{1}{2}$, see (6.2).

Thresholds

Even when we apply condensation, we still have a decrease in correlation values for high values of `#unjudged`. One way of dealing with this problem is to choose an optimal

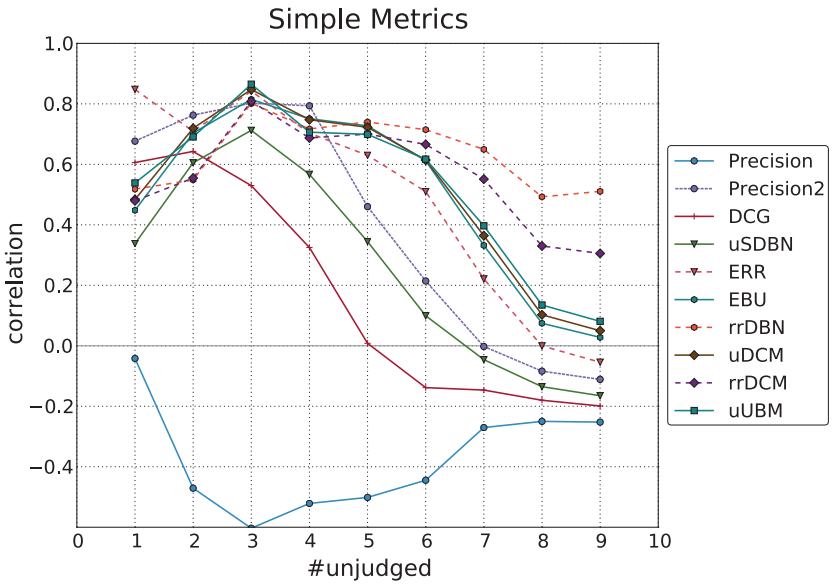


Figure 6.1: Pearson correlation between offline metrics and *interleaving signal*. Unjudged documents are treated as irrelevant.

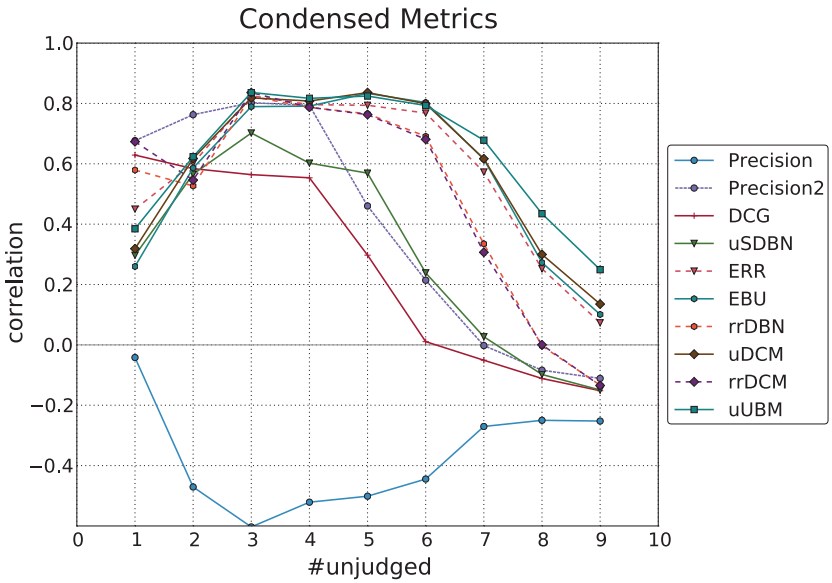


Figure 6.2: Pearson correlation between offline metrics and *interleaving signal*. Unjudged documents are skipped (ranked lists are condensed).

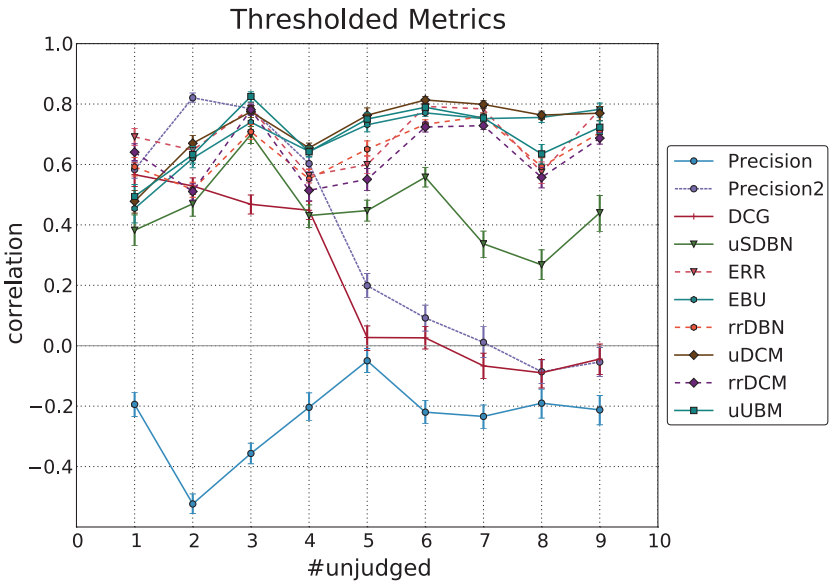


Figure 6.3: Pearson correlation between offline metrics with thresholds and *interleaving signal*. Unjudged documents are treated as irrelevant.

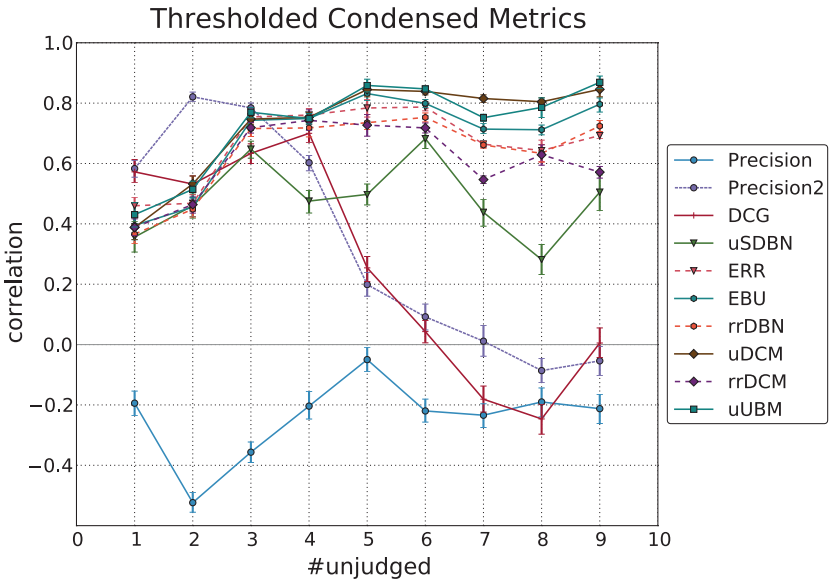


Figure 6.4: Pearson correlation between offline metrics with thresholds and *interleaving signal*. Unjudged documents are skipped (ranked lists are condensed).

value of `#unjudged` and use it to get high correlations with interleaving outcomes. We propose a different way of dealing with this noisy data. Comparing systems A and B, we discard all queries with differences in metric values less than a threshold δ_m for each metric m:

$$MetricSignal = \frac{1}{|Q_{\delta_m}|} \sum_{q \in Q_{\delta_m}} \left(m(B,q) - m(A,q)\right),$$

where $Q_{\delta_m} = \{q \in Q \mid |m(B,q) - m(A,q)| \geqslant \delta_m\}$. This means that we use only some portion of the queries we have (up to 20%), but these are queries that strongly distinguish between systems. The idea is that by choosing an appropriate threshold δ_m we can tune a ranking system to produce the best correlation with interleaving outcomes. In order to test the idea we split our data (ten TDI experiments) into train and test set: we use the train set to choose the best threshold and the test set to compute the correlation scores.

While it would be natural to do a time-based train/test split, it appeared to be impractical with the data we have. Firstly, it turns out to be impossible to get training and test sets of reasonable sizes (either the training or the test set would consist of only three experiments which might give too noisy correlation values). Secondly, there are only few possible time-based splits so we are not able to assess statistical significance of the results. Instead, we use all possible 5/5 splits for our experiments, i.e., we take a subset of five experiments as a training set and the remaining five experiments as a test set. In total, we have $C_{10}^5 = 252$ splits and corresponding correlation values. The correlation values are then averaged and error bars are computed using the bootstrap test [71] at 95% confidence level and $1{,}000$ samples. Results are shown in Figures 6.3 and 6.4.

We can see in Figures 6.3 and 6.4 that the confidence intervals are quite narrow and most of the click model-based metrics continue to show high correlation scores as the value `#unjudged` increases. If we look at one of the best performing metrics, uUBM, we can see that thresholded variants are a bit worse for `#unjudged` lower than 5, while for `#unjudged` equal to 5 and higher the thresholded variants start dominating, reaching the highest point for `#unjudged` $= 9$ (see Figure 6.5).

In order to test significance of the differences in correlation values we use the 5/5 split procedure described above. Unlike what we do for *thresholded* and *thresholded condensed*, for the *simple* and *condensed* variants we only use the test set to determine the correlation and just ignore the training set as there is nothing we need to tune. The correlation values are then averaged and confidence intervals are computed using the bootstrap method with $1{,}000$ samples and 95% confidence level. The three highest correlation scores were shown by a *thresholded condensed* variant of uUBM metric (for different values of `#unjudged`), while the correlation score for *thresholded condensed* uUBM (`#unjudged` $= 9$) is significantly higher than any other variant (*simple*, *condensed*, *thresholded*) of any metric. From Figures 6.1–6.4 we can also conclude that click model-based metrics in general show higher correlation values with the outcomes of interleaving experiments than traditional offline metrics, especially when we have many incomplete judgements (`#unjudged` > 5), which confirms the hypothesis formulated in the introduction: *click model-based* metrics are better correlated with online measurements than traditional metrics. Another interesting observation is that for the *simple* and *condensed* variants there exist optimal values of the `#unjudged` parameter (3 and 5 respectively in our case). Conversely,

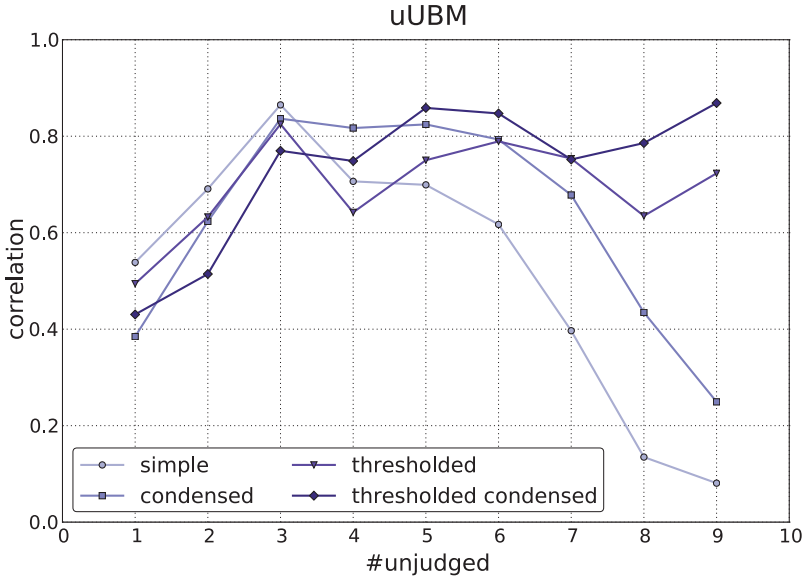


Figure 6.5: Pearson correlation between uUBM (in different variants) and *interleaving signal*.

for the *thresholded* and *thresholded condensed* variants it is more important to pick an appropriate metric and then use any value of `#unjudged` higher than five.

6.4.2 Correlation with Absolute Online Metrics

Following the original work on ERR by Chapelle et al. [26] we also compare offline IR metrics by looking at their correlation with absolute click metrics. In our experiments we use Max-, Min- and MeanRR, UCTR and PLC metrics (see Section 3.3). We do not include the search success (SS) metric considered by Chapelle et al. [26] as it uses relevance labels and not only clicks. We have also confirmed the findings of [26] that QCTR (clicks per session) has negative or close to zero correlation with all the editorial metrics and therefore skip it as well.

A *configuration* is a tuple that consists of a query and ten URLs of the top ranked documents presented to a user. For each configuration in our dataset we compute the values of absolute online and offline metrics. The vectors of these metric values are then used to compute Pearson correlation (unweighted). For our dataset we use clicks collected during a three-month period in 2012. Because we use a long period and hence have a sufficient amount of data, we are able to collect 12,155 configurations (corresponding to 411 unique queries) where all ten documents have relevance judgements.

The results are summarized in Table 6.2. A similar comparison was previously done by Chapelle et al. [26] for ERR and traditional offline metrics. The numbers they obtained are similar to ours. From the table we conclude that click model-based metrics show

Table 6.2: Pearson correlation between offline and absolute online metrics. Superscripts denote cases of statistically significant difference with ERR and EBU respectively. The first (second) ▲ means that the metric is statistically significantly higher than ERR (EBU), ▼—significantly lower, ◇—no statistical difference can be found (95% significance level, bootstrap test).

	-RR				
	Max-	Min-	Mean-	UCTR	PLC
Precision	−0.117	−0.163	−0.155	0.042	−0.027
Precision2	0.026	0.093	0.075	0.092	0.094
DCG	0.178	0.243	0.237	0.163	0.245
ERR	0.378	0.471	0.469	0.199	0.399
EBU	0.374	0.467	0.464	0.198	0.397
rrDBN	$0.384^{\blacktriangle\blacktriangle}$	$0.475^{\blacktriangle\blacktriangle}$	$0.473^{\blacktriangle\blacktriangle}$	$0.194^{\blacktriangledown\blacktriangledown}$	$0.399^{\diamond\blacktriangle}$
rrDCM	$0.387^{\blacktriangle\blacktriangle}$	$0.478^{\blacktriangle\blacktriangle}$	$0.476^{\blacktriangle\blacktriangle}$	$0.194^{\blacktriangledown\blacktriangledown}$	$0.400^{\diamond\blacktriangle}$
uSDBN	$0.322^{\blacktriangledown\blacktriangledown}$	$0.412^{\blacktriangledown\blacktriangledown}$	$0.407^{\blacktriangledown\blacktriangledown}$	$0.206^{\blacktriangle\blacktriangle}$	$0.370^{\blacktriangledown\blacktriangledown}$
uDCM	$0.374^{\blacktriangledown\blacktriangledown}$	$0.466^{\blacktriangledown\blacktriangledown}$	$0.463^{\blacktriangledown\blacktriangledown}$	$0.198^{\diamond\diamond}$	$0.396^{\blacktriangledown\blacktriangledown}$
uUBM	$0.377^{\diamond\blacktriangle}$	$0.469^{\blacktriangledown\blacktriangle}$	$0.467^{\blacktriangledown\blacktriangle}$	$0.198^{\diamond\diamond}$	$0.398^{\diamond\blacktriangle}$

relatively high correlation scores while traditional offline metrics like DCG or Precision generally have lower correlations, which agrees with the results of the previous section. Using the bootstrap test (95% significance level, 1,000 bootstrap samples) we confirmed that all the click model-based metrics show significantly higher correlation with all the online metrics than any of the traditional offline metrics.

As to the online metrics, we can see that the reciprocal rank family (MaxRR, MinRR, MeanRR) appears to be better correlated with the *effort-based* metrics (ERR, rrDBN, rrDCM), because the effort function used by these metrics is the reciprocal rank $\frac{1}{r}$ (6.2). The same holds for PLC as it uses reciprocal rank of the lowest click that could be viewed as "satisfaction position" used by an effort-based metric. The differences between ERR and uSDBN, rrDBN and EBU, rrDCM and uDCM are statistically significant (using the same bootstrap test). Conversely, for the UCTR metric all the utility-based metrics show significantly higher correlation than corresponding effort-based metrics.

We also compare the newly introduced click model-based metrics with older metrics: ERR (effort-based) and EBU (utility-based). The result of the comparison is marked as superscripts in the Table 6.2: the first superscript corresponds to ERR, the second one corresponds to EBU. As we see, in most cases our new click metrics appear to be significantly better than the previously known ERR and EBU metrics, expect for UCTR measure, which does not account for clicks (rather for their absence) and hence obviously lacks the source of correlation with click-model based metrics. According to other metrics, rrDBN and rrDCM are better than ERR in three out of four cases and better than EBU in all four cases, while uUBM is better than EBU in four out of four cases.

In general, all the absolute click metrics are poorly correlated with offline metrics—the correlation values are much lower than correlation with interleaving outcomes. As

Table 6.3: Correlation between offline metrics (using the TREC 2011 runs). Values higher than 0.9 are marked in boldface.

	Precision2	DCG	ERR	uSDBN	EBU	rrDBN	uDCM	rrDCM	uUBM
Precision	0.649	0.841	0.597	0.730	0.568	0.397	0.562	0.442	0.537
Precision2	–	0.785	0.663	0.780	0.675	0.526	0.693	0.551	0.681
DCG	–	–	0.740	0.857	0.711	0.530	0.704	0.592	0.685
ERR	–	–	–	0.807	**0.919**	0.754	**0.902**	0.826	0.888
uSDBN	–	–	–	–	0.792	0.585	0.794	0.638	0.754
EBU	–	–	–	–	–	0.788	**0.970**	0.822	**0.930**
rrDBN	–	–	–	–	–	–	0.786	**0.917**	0.807
uDCM	–	–	–	–	–	–	–	0.813	**0.947**
rrDCM	–	–	–	–	–	–	–	–	0.841

was shown by Radlinski et al. [127], absolute click metrics are worse at capturing user satisfaction than interleaving. That is why we propose to use the results of Section 6.4.1 as the main way to compare offline metrics with user behavior.

6.4.3 Correlation Between Offline Metrics

In order to compare offline metrics to each other in terms of ranking IR systems we use data from the TREC 2011 Web Track [54]. Participants of the TREC competition were offered a set of queries ("topics" in TREC parlance) and a set of documents for each query to rank. Each document was judged using a four-grade scale.[6] For each metric we can build a list of system runs[7] ordered by the metric value averaged over queries. We then compute Kendall tau correlation scores between these ordered lists; they are summarized in Table 6.3. As was shown by Voorhees [158], metrics with correlation scores around 0.9 can be treated as very similar because this is the level of correlation one achieves when using the same metric but different judges. This level of correlation to distinguish equivalent metrics was also used in subsequent papers, for example [18, 22, 135, 157]. In Table 6.3 such metric pairs are marked in boldface.

We see that all click model-based metrics are highly correlated within their group, *utility-based* or *effort-based*, while correlations of the two metrics based on the same model (uSDBN and ERR, EBU and rrDBN, uDCM and rrDCM) are lower.

6.4.4 Discriminative Power

Another measure frequently used for comparing metrics is the *discriminative power* by Sakai [130]. This measure is a bit controversial, because high values of discriminative power do not imply a good metric. Nevertheless, extremely low values of discriminative

[6] Initially, a five-grade scale was listed on a TREC 2011 description page, but in the end a four-grade scale was used for evaluation. As in the `trec_eval` evaluation tool we do not distinguish between *Irrelevant* and *Spam* documents.

[7] In total we have 62 runs submitted by 16 teams.

Table 6.4: Discriminative power of different metrics according to the bootstrap test (confidence level 95%).

Metric	Discriminative Power
Precision	50.1%
Precision2	30.8%
DCG	48.6%
ERR	39.3%
uSDBN	51.1%
EBU	35.1%
rrDBN	21.1%
uDCM	34.7%
rrDCM	26.0%
uUBM	33.3%

power can serve as an indication of a metric's poor ability to distinguish different rankings. As was shown in previous work (e.g., [53, 146]) discriminative power is highly consistent with respect to statistical test choice. Given this fact we focus on a bootstrap test as it makes fewer assumptions about the underlying distribution. Results based on the same TREC 2011 Web Track data as used in the previous section are summarized in Table 6.4. As expected, highly correlated metric pairs (e.g., (rrDBN, rrDCM) and (EBU, uDCM)) have similar discriminative power.

Another observation to be made is that the *effort-based* metrics ERR, rrDBN and rrDCM have a lower discriminative power than the *utility-based* metrics uSDBN, EBU and uDCM, respectively. This is probably due to the fact that "position discount" for the effort-based metrics goes to zero faster than for the utility-based metrics and hence they are less sensitive to changes in the bottom of the ranked list.

6.5 Conclusion

In this chapter we proposed a framework of *click model-based* metrics to build an offline evaluation measure on top of any click model.

We formulated the following research questions:

RQ3.1 Can we make use of click models to build better evaluation metrics? How do such click model-based IR metrics differ from traditional offline metrics?

RQ3.2 Which evaluation metrics are better tied to the user? Do click model-based metrics show higher agreement with online experiments? How do they compare in terms of discriminative power?

RQ3.3 How well do different offline metrics perform in the presence of unjudged documents?

RQ3.4 How can we modify offline metrics to enhance agreement with online experiments?

Answering them, we can say the following:

- *Click model-based* metrics generally differ from traditional offline metrics, while they are quite similar to each other. Moreover, *utility-based* metrics are significantly different from *effort-based* metrics in terms of system ranking.
- All *click model-based* metrics generally show high agreement with the outcomes of online interleaving experiments and relatively high agreement with absolute click measures. However, correlation with absolute metrics is low for all offline metrics (both traditional and click model-based) compared to the correlation with interleaving outcomes.
- Unjudged documents may decrease correlation with interleaving outcomes but by using thresholds we can overcome this issue for click model-based metrics.
- *Condensation* and *thresholding* of offline metrics are effective ways of stabilizing correlations with interleaving outcomes in the presence of unjudged documents.

Future directions. Naturally, the study we performed in this chapter has some limitations and potential for future work. Adapting click models for unjudged/unknown documents is one such direction. For example, one could modify a click model by adding the probability of a document being skipped because it is unjudged. This question requires further investigation and we leave it as future work.

Another natural extension of our framework of *click model-based* metrics is adding more signals from the assessors. For instance, we can ask assessors to judge not only documents, but their snippets as well (a practice already in place at commercial search engines) and use direct snippet relevance as part of the total utility.

Finally, in our work we argued that offline metrics should be better correlated with interleaving outcomes. However, we might want to have a metric that correlates with self-reported user satisfaction instead.

The last two topics will be addressed next, in Chapter 7. Alternatively, the reader may proceed to Chapter 8 which addresses the question of *online* evaluation as opposed to offline evaluation studied in the current and the following chapters.

7

Offline Evaluation of Modern Search: Click, Attention and Satisfaction

Modern SERPs often provide immediate value to users and organize information in such a way that it is easy to navigate. The core ranking function contributes to this and so do result snippets, smart organization of result blocks and extensive use of one-box answers or side panels. While they are useful to the user and help search engines to stand out, such features present two big challenges for evaluation. First, the presence of such elements on a SERP may lead to the absence of clicks, which is, however, not related to dissatisfaction, so-called "good abandonments." Second, the non-linear layout and visual difference of SERP items may lead to non-trivial patterns of user attention, which is not captured by existing evaluation metrics.

In this chapter we propose a model of user behavior on a SERP that jointly captures click behavior, user attention and satisfaction, the *clicks, attention and satisfaction* (CAS) model, and demonstrate that it gives more accurate predictions of user actions and self-reported satisfaction than existing models based on clicks alone. We use the CAS model to build a novel evaluation metric that can be applied to non-linear SERP layouts and that can account for the utility that users obtain directly on a SERP. We demonstrate that this metric shows better agreement with user-reported satisfaction than conventional evaluation metrics.

Together these contributions allow us to answer the research questions formulated in the introduction:

RQ4.1 Does a model that unites attention and click signals give more precise estimations of user behavior on a SERP and self-reported satisfaction? How well does the model predict click vs. satisfaction events?

RQ4.2 Does an offline evaluation metric based on such a model show higher agreement with user-reported satisfaction than conventional metrics such as DCG?

7.1 Introduction

When looking at the spectrum of queries submitted to a web search engine, we see a heavy head of highly frequent queries ("head queries") as well as a long tail of low-frequency

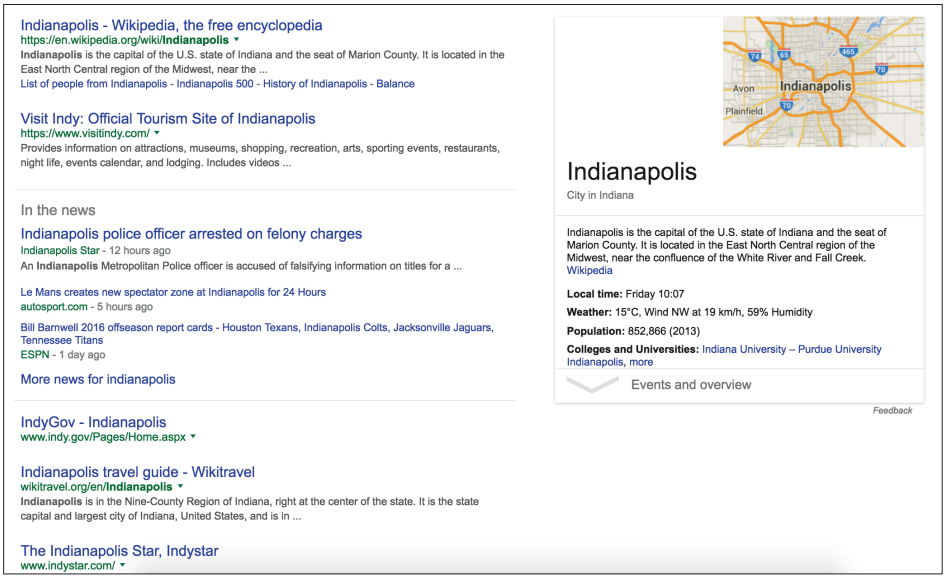


Figure 7.1: Example of a modern SERP with a news block and a side panel produced by one of the big commercial search engines for the query `[Indianapolis]`. Image credits: Google.

queries ("tail queries") [145]. While a small number of head queries represent a big part of a search engine's traffic, all modern search engines can answer these queries quite well. In contrast, tail queries are more challenging, and improving the quality of results returned for tail queries may help a search engine to distinguish itself from its competitors. These queries often have an underlying *informational* user need: it is not the user's goal to navigate to a particular website, but rather to find out some information or check a fact. Since the user is looking for information, they may well be satisfied by the answer if it is presented directly on a SERP, be it inside an information panel or just as part of a good result snippet. In fact, as has been shown by Stamou and Efthimiadis [151], a big portion of abandoned searches is due to pre-determined behavior: users come to a search engine with a prior intention to find an answer on a SERP. This is especially true when considering mobile search where the network connection may be slow or the user interface may be less convenient to use.

An important challenge arising from modern SERP layouts is that their elements are visually different and not necessarily placed in a single column. As was shown by Dumais et al. [68], grouping similar documents helps user to navigate faster. Since then this approach has been studied extensively by the IR community [5–7, 122, 171] and adopted by the major search engines with so-called vertical blocks and side panels (Figure 7.1). When information is presented in such a way, the user examines it in a complex way, not by simply scanning it from top to bottom [62, 159, 160].

We claim that the currently used user models and corresponding evaluation metrics

have several disadvantages. First, most of the models assume that the SERP consists of equally shaped result blocks, often homogeneous, presented in one column, which often prevents us from accurately measuring user attention. Second, none of the current Cranfield-style evaluation metrics account for the fact that the user may gain utility directly from the SERP. And finally, and, perhaps, the most important of all, is that the offline evaluation metrics, although sometimes based on a user model, do not learn from the user-reported satisfaction, but rather use ad-hoc notions for utility and effort.

In this chapter we propose an offline evaluation metric that accounts for non-trivial attention patterns of modern SERPs and the fact that a user can gain utility not only by clicking documents, but also by simply viewing SERP items. Our approach consists of two steps, each having value on its own: (1) we build a unified model of a user's clicks, attention and satisfaction, the *clicks, attention and satisfaction* (CAS) model; and (2) we use this model to build a Cranfield-style evaluation metric (which we call the CAS metric).

The rest of the chapter is organized as follows. In Section 7.2 we discuss related work. In Section 7.3 we present our motivation for including utility gained directly from the SERP. Then we present our user model in Section 7.4. In Section 7.5 we present an evaluation metric based on this model. Section 7.6 describes our experimental setup. In Section 7.7 we present results of our experiments. We conclude in Section 7.8.

7.2 Related Work

Apart from click models discussed in Section 2.3 and click model-based metrics introduced in Chapter 6, there are three areas of research relevant to the current study. Below we list the most relevant papers from each of them.

Abandonments

Turpin et al. [157] showed that perceived relevance of the search results as seen on a SERP (snippet relevance or direct SERP item relevance, as we call it) can be different from the actual relevance and should affect the way we compute the utility of a page. Li et al. [104] introduced the notion of *good abandonment* showing that utility can be gained directly from the SERP without clicks. Chilton and Teevan [30] found that the presence of specially decorated search results (*Answers*) might lead to higher abandonment rates for several query types. In [38, 39] two approaches to query classification were suggested, where the queries were classified as *potential good abandonments*, i.e., the queries that are likely to result in good abandonment. Diriye et al. [63] presented a comprehensive analysis of causes of (good and bad) abandonments, while Williams et al. [161] and Song et al. [147] showed how to build a classifier of abandonments.

Mouse Movements

Another important part of related studies concerns *mouse movements*. It was demonstrated that there is a strong relation between mouse movement and eye fixation, although this relation is not trivial [129]. Even though the correlation between eye fixation and mouse movement is far from perfect, the latter was shown to be a good indicator of user

attention [118], comparable in quality to eye gaze data. In follow-up work Navalpakkam et al. [119] showed that mouse movements are not always aligned with eye fixations, suggesting the idea that this behavior is user-dependent. Based on the idea of eye-mouse association, a classifier was developed that can predict the fact of an individual user carefully reading a SERP item [108] and even the satisfaction reported by the user [109], based on mouse movements. Huang et al. [87] demonstrated that mouse movements can serve as a strong signal in identifying good abandonments. Diriye et al. [63] showed that mouse movement data together with other signals can indeed yield an efficient classifier of good abandonments. Their work also introduced an experimental setup for in-situ collection of good abandonment judgements. They argued that this is the only way of collecting ground truth data, as even query owners have difficulties telling the reason for abandonment if they are asked later.

Click Models

On the overlap between click modeling and mouse movement research Huang et al. [88] proposed an extended click model that uses mouse interactions to slightly refine an existing click model. Diaz et al. [62] showed that visually salient SERP elements can dramatically change mouse movement trails and suggested a model that handles this. Chen and Min [29] adopted a generative approach to click modeling where relevance, clicks and mousing were written as noisy functions of previous user actions. They used this approach to predict CTRs of results on the SERP.

Our work here is different from previous work on good abandonments in that we not only allow for stopping after a good SERP item, but we account for this in terms of the total utility accumulated by the user, which brings us closer to the traditional Cranfield-style evaluation approach. Our work is different from previous work on mouse movements and click models in that we do not study them separately, but use both as evidence for locating the user's attention. On top of that, we explicitly include in our model the notion of accumulated utility and user satisfaction as well as the possibility to gain utility from results that were not interacted with.

7.3 Motivation

In this section we demonstrate that the presence of snippets that answer the user query increases the number of abandonments. That suggests that the user can be satisfied without a click. A similar study for mobile search was carried out by Arkhipova and Grauer [9], who performed online experiments and demonstrated that satisfaction might come from snippets, not just from clicked results.

Chapelle et al. [26] showed that conventional offline metrics such as DCG (Section 3.2.4) have negative correlation with abandonment rate.[1] However, the phenomenon of *good abandonments* [104] must be taken into account. In all previous works related to good abandonments authors usually performed human assessments of query-SERP pairs by asking human raters the following question: "Does the search engine result page

[1] They showed a positive correlation with UCTR, which is the opposite of abandonment (Section 3.3).

p contain an answer to the query q?" Still, it is not completely clear if real users (i.e., not raters) notice such an answer and decide to abandon their habit of clicking on search results. In this section we measure correlation of snippet metrics with user clicks thus providing evidence for real-word good abandonments.

For this small motivation study we focus on factoid queries: short questions that can be answered in a search result snippet (annotation). Our idea is to gather judgements of snippet quality, compute offline metrics that measure a SERP's quality and then calculate correlation with click metrics. We start by extracting data from Yandex's query log. We keep queries with clear encyclopedic intent, namely queries of the form [X definition], [what is X], [meaning of X], etc. After that we ask a group of raters to judge each document snippet with respect to the query:

- Raters should answer *Yes* if it represents a full and easy-to-read answer to the question.
- Raters should answer *No* if it does not contain an answer to the user's query (even if an acceptable answer can be found after following the URL).
- Raters should answer *Partial* if it contains only partial answer to the query. E.g., the question is to define some ambiguous term and the snippet explains only one of its possible meanings. Another example: answer is correct, but not detailed enough or is not easy to read.

After performing this filtering, we are left with $8{,}830$ judged snippets corresponding to 883 queries. In order to calculate whole SERP relevance we compute several frequently used offline metrics (see Sections 3.2 and 6.3.1):[2]

- **Precision at N**. We calculate P@10 and P@5 by converting relevance grades to binary (only *Yes*-snippets are treated as relevant).
- **Average precision (AP)**. Again, only *Yes*-snippets are treated as relevant.
- **uSDBN**. We use uSDBN metric (6.6) also known as "ERR with abandonment probability" [26, Section 7.2]: We convert direct relevance grades D to probabilities using the mapping: $\rho(D) = \frac{2^D-1}{2^{D_{max}}}$, as in Section 3.2.4. In our case $D \in \{0, 1, 2\}$ is a grade assigned by a rater, $D_{max} = 2$. Similar to Section 6.3.1, we use $\gamma = 0.9$ as was suggested in [25].
- **Cumulative gain (CG)**. Non-discounted metric: $\sum_{r=1}^{n} \rho(D_r)$.
- **Discounted cumulative gain (DCG)**. Classic DCG metric: $\sum_{r=1}^{n} \frac{2^{D_r}-1}{\log_2(1+r)}$.

Once we have computed quality measures for all the SERPs we compare them to online click-based metrics.[3] The hypothesis is that users tend to click less when SERP is sufficiently good. First, we perform the following procedure: for each offline metric we sort queries according to this metric (in ascending order) and split our data set into five

[2] We replace document relevance R by direct snippet relevance D.

[3] Clicks are gathered from a Yandex query log collected for the period of three months in 2011. In total, we have $137{,}010$ query sessions.

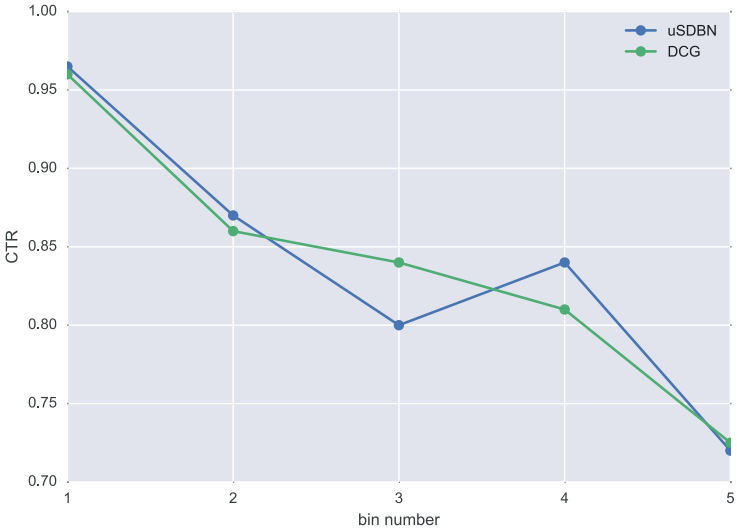


Figure 7.2: Pearson correlation between CTR and SERP quality.

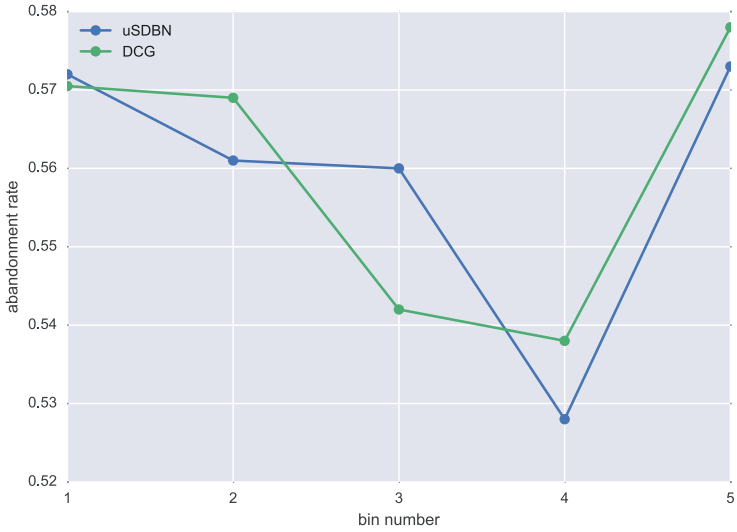


Figure 7.3: Pearson correlation between abandonment rate and SERP quality.

Table 7.1: Pearson correlation between SERP quality metrics and abandonment rate.

	P@10	P@5	AP	CG@10	CG@5	DCG	uSDBN
CTR	-0.154	-0.142	-0.111	-0.190	-0.182	-0.195	-0.186
Aband. (whole dataset)	-0.010	-0.003	-0.041	$+0.023$	$+0.022$	$+0.012$	-0.021
Aband. (best 40%)	$+0.150$	$+0.074$	$+0.006$	$+0.114$	$+0.084$	$+0.084$	$+0.078$

bins of equal size, each bin contains one fifth of all instances. For that experiment we use only two offline metrics: uSDBN and DCG. Other metrics are too discrete and hard to split to equal bins. We examined two click metrics, namely *abandonment rate* and *total CTR*, (see Section 3.3). The results are presented in Figures 7.2 and 7.3.

From these plots we can draw several conclusions. First, we can see that overall page CTR decreases with better snippets' quality, i.e., better SERPs need fewer clicks. Second, we conclude that users tend to abandon more the SERPs that contain very informative snippets.

We then look at the CTR metric for the whole dataset and find substantial negative correlation with all SERP quality metrics. For abandonment rate we compute correlation for the whole dataset and also use another approach where we leave only well-performing queries, i.e., best 40% of the queries according to a particular offline metric. The intuition is that those queries are more likely to result in good abandonments and we expect to see positive correlation with abandonment rate, whereas it could be negative for the whole dataset. The results are summarized in Table 7.1 and, indeed, confirm our intuition.

Results presented in this section can be considered as a justification of the existence of real *good abandonments* in the real Web. Moreover, we claim that asking the raters to judge the snippets for *direct snippet relevance* is useful in determining user satisfaction. In the next section we take this motivation on board and present a new user model and a corresponding evaluation metric.

7.4 Model

Let us first describe the *clicks, attention and satisfaction* (CAS) model that we are going to use. It is a model of user behavior on a SERP that has three components:

- an attention model;
- a click model; and
- a satisfaction model.

The model is visualized in Figure 7.4. Each SERP item k gives rise to a feature vector $\vec{\varphi}_k$ that determines the examination event E_k. After examination the user may or may not click through (C_k). Then the examined and clicked documents contribute to the total utility, which, in turn, determines satisfaction (S). We describe each of the three components in the following sections.

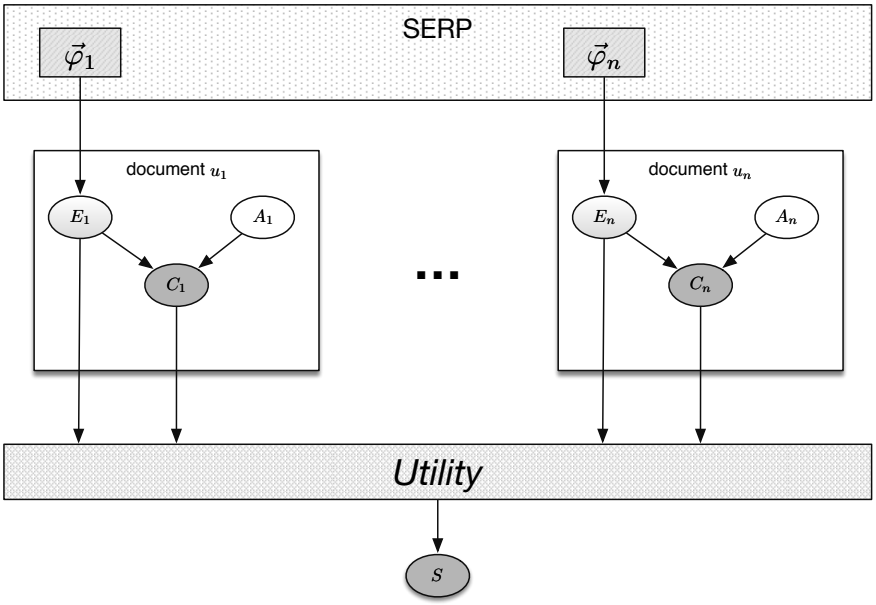


Figure 7.4: Diagram of the CAS model.

We should note here that we train a *relevance-based* click model, where the click probability depends on the relevance label assigned by the raters and not on the document itself, just as we did in Chapter 6. A classical click model can also be trained (from a bigger dataset) and compared using click likelihood similar to what we did in Chapter 5. However, we still need a relevance-based model to build an evaluation metric (Sections 7.5 and 7.7.2).

7.4.1 Attention (Examination) Model

Diaz et al. [62] suggested a model that predicts mouse transitions between different elements of the SERP. While mousing can be used as a proxy for user attention focus [77, 119, 129], we observe in our data entire classes of sessions where mouse tracks and attention areas are substantially different, while others are not.[4] Hence, we cannot fully reconstruct the attention transition path. That is why, unlike [62], we train a *pointwise* model of user attention:

$$P(E_k = 1) = \varepsilon(\vec{\varphi}_k), \tag{7.1}$$

where k is an index referring to one of the items comprising the SERP (result snippets, weather results, knowledge panels, etc.), E_k is a random variable corresponding to the user examining item k, $\vec{\varphi}_k$ is a vector of features indexed by the item k, and ε is a

[4]For instance, currency conversion queries like, e.g., [12 EUR in CHF] often result in no mousing at all, yet the user reports satisfaction. Similar patterns of discrepancy between mousing and attention were also reported by Rodden et al. [129].

function converting a feature vector into a probability. The features we use are presented in Table 7.2.

Table 7.2: Features used by the attention model of CAS.

Feature group	Features	# of features
rank	user-perceived rank of the SERP item (can be different from k)	1
CSS classes	SERP item type (Web, News, Weather, Currency, Knowledge Panel, etc.)	10
geometry	offset from the top, first or second column (binary), width (w), height (h), $w \times h$	5

The function that converts feature vectors into probabilities is a logistic regression. Instead of training it directly from mouse movement data, which is only a part of the examined items, we train it in such a way that it optimizes the full likelihood of the data, which includes not just mouse movement, but also clicks and satisfaction labels. More on this in the following sections.

7.4.2 Click Model

For our click model we use a generalization of the *position-based model* (PBM) (Section 3.1.2), at the core of which lies an examination hypothesis, stating that in order to be clicked a document has to be examined and attractive:

$$P(C_k = 1 \mid E_k = 0) = 0 \tag{7.2}$$
$$P(C_k = 1 \mid E_k = 1) = \alpha_{u_k}, \tag{7.3}$$

where C_k is a random variable corresponding to clicking the k-th SERP item, α_{u_k} is the attractiveness probability of the SERP item u_k. Unlike the classic PBM model, where examination is determined by the rank of the SERP item, in our model we use a more general approach to compute the examination probability $P(E_k = 1)$, as described in Section 7.4.1.

7.4.3 Satisfaction Model

Next, we propose a satisfaction model. As we saw in Section 7.3 user satisfaction may come from clicking a relevant result, but also from examining a good SERP item. We also assume that satisfaction is not a binary event that happens during the query session, but has a cumulative nature. In particular, we allow the situations where after examining a good document or a good SERP item the user may still continue the session. This assumption is supported by data that we collected from raters. After looking at a SERP item (referred to as "summary extracted from a bigger document" in the instructions), our raters were asked whether they think that "examining the full document will be useful to answer the question Q" and if so, what the reason is. While looking at the reasons specified by the raters we found out that 42% of the raters who said that they would click

through on a SERP, indicated that their goal was "to *confirm* information already present in the summary," which implies that the summary has an answer, yet the users continue examining it.

To put these ideas into a model, we assume that each relevant document or SERP item that received a user's attention contributes towards the total utility U gained by the user:

$$U = \sum_k P(E_k = 1) u_D(\vec{D}_k) + \sum_k P(C_k = 1) u_R(\vec{R}_k), \tag{7.4}$$

where $\vec{D}_k$ and $\vec{R}_k$ are vectors of rater-assigned labels of direct SERP item relevance and full document relevance, respectively; u_D and u_R are the transformation functions that convert the corresponding raters' labels into utility values. To accommodate variable ratings from different raters, we assume u_D and u_R to be linear functions of the rating histogram with weights learned from the data:

$$u_D(\vec{D}_k) = \vec{\tau}_D \cdot \vec{D}_k \tag{7.5}$$

$$u_R(\vec{R}_k) = \vec{\tau}_R \cdot \vec{R}_k, \tag{7.6}$$

where $\vec{D}_k$ and $\vec{R}_k$ are assumed to be histograms of the ratings assigned by the raters. We have three grades for D (see Figure 7.7, question 2) and four relevance grades for R (*Irrelevant*, *Marginally Relevant*, *Relevant*, *Perfect Match*); the vectors have corresponding dimensions.

Then, we assume that the probability of satisfaction depends on the accumulated utility via the logit function:

$$P(S = 1) = \sigma(\tau_0 + U) = \frac{1}{1 + e^{-\tau_0 - U}}, \tag{7.7}$$

where τ_0 is a constant intercept value.

Finally, we can write down the satisfaction probability as follows:

$$P(S = 1) = \sigma\left(\tau_0 + \sum_k P(E_k = 1) u_D(\vec{D}_k) + \sum_k P(C_k = 1) u_R(\vec{R}_k)\right). \tag{7.8}$$

7.4.4 Model Training

To be able to train the CAS model we make a further assumption that the attractiveness probability α_{u_k} depends only on the relevance ratings $\vec{R}_k$ assigned by the raters:[5]

$$P(C_k = 1 \mid E_k = 1) = \alpha(\vec{R}_k) = \sigma\left(\alpha^0 + \vec{\alpha} \cdot \vec{R}_k\right). \tag{7.9}$$

Since the function α has to yield a probability, we set it to be a logistic regression of the rating distribution.

Now that we have the model fully specified, we can write the likelihood of the observed mouse movement, click and satisfaction data and optimize it using a gradient descent

[5] We also tried collecting separate attractiveness labels with the crowd worker, but the data appeared to be too noisy due to the subjective nature of the question; see Section 7.6.2 for more details.

method. We use the L-BFGS algorithm [107], which is often used for logistic regression optimization. It has also been shown to be robust to correlated features [114].

One important thing to note is that while computing the satisfaction probability (7.8) as part of the likelihood expression, the values of click probabilities are always either 0 or 1, while the value of the examination probability can be either 1 if there is a mouse fixation or it is computed using (7.1) if there is no mouse fixation on the SERP item.

7.5 Search Evaluation Metric

Now that we have described a model of the user's behavior on a SERP, we can use this model to build an evaluation metric. Once the parameters of the model are fixed, it can easily be re-used for any new search ranking or layout change. This is very important when working on improving a search engine and allows for quick iterations.

Assume that we have the following judgements about the SERP items from human raters (cf. [33]):

1. direct SERP item relevance D_k; and
2. topical relevance R_k of the full document (assigned after clicking and examining the full document).

Assume further that we have trained the model as explained in Section 7.4.4. Now we can simply plug in the relevance labels and the model parameters in equation (7.4) to obtain the utility metric:

$$U = \sum_k \varepsilon(\vec{\varphi}_k) \left(u_D(\vec{D}_k) + \alpha(\vec{R}_k) u_R(\vec{R}_k) \right). \tag{7.10}$$

Note that after the parameters have been estimated and fixed, only the raters' judgements and layout information are used to evaluate system performance. This way we ensure the scalability and re-usability of the Cranfield-style offline evaluation.

7.6 Experimental Setup

First, let us repeat the research questions for this chapter:

RQ4.1 Does a model that unites attention and click signals give more precise estimations of user behavior on a SERP and self-reported satisfaction? How well does the model predict click vs. satisfaction events?

RQ4.2 Does an offline evaluation metric based on such a model show higher agreement with user-reported satisfaction than conventional metrics such as DCG?

Our first research question **RQ4.1** requires that we build a model and evaluate it on self-reported satisfaction. That prompts us to collect a log of user actions. See Section 7.6.1. Similarly, for the second question **RQ4.2** we need to have judgements from independent raters and we use crowdsourcing for it. See Section 7.6.2.

Below we carefully describe each step of our data collection to facilitate reproducibility. Then we detail the baseline models and the way we evaluate the models.

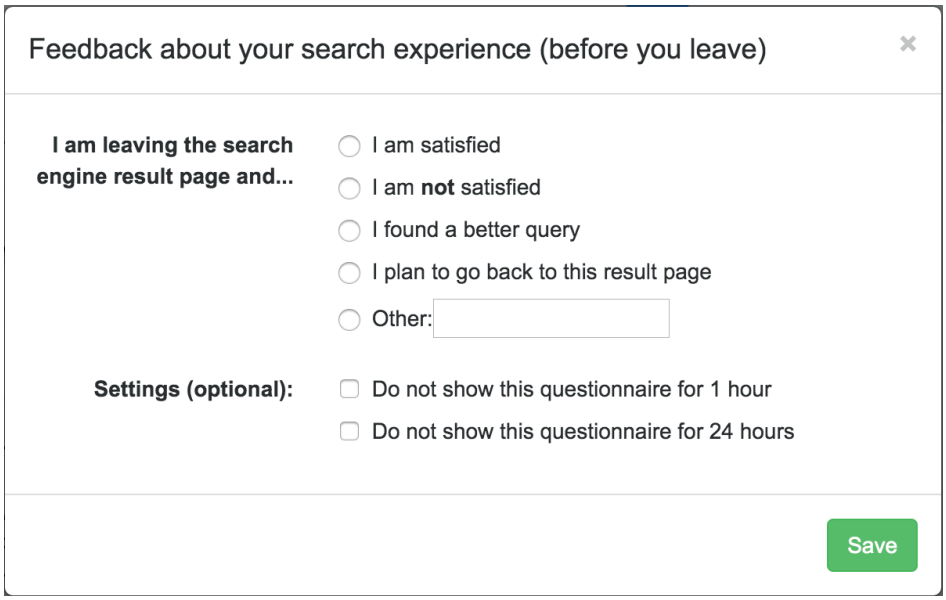


Figure 7.5: Search satisfaction questionnaire.

7.6.1 In-situ Data Collection

First of all, we set up a proxy search interface that intercepts user queries to a commercial search engine and collects click and mouse movements data. Our log collection code is based on the `EMU.js` library by Guo and Agichtein [76]. The interface was used by a group of volunteers who agreed to donate their interaction data. The design of the experiment was also reviewed by the Ethical Committee of the University of Amsterdam. In our experiments we only use the queries that were explicitly vetted by the owners as not privacy sensitive using the log management interface we provide; see Figure 7.6.[6] We should also stress here, that unlike laboratory settings, the search experience was not changed: the user received the same list of results and interacted with them in the same way as if they were using the underlying search system in the normal manner.

Occasionally we showed a pop-up questionnaire asking users to rate their search experience upon leaving the SERP; see Figure 7.5. To avoid showing it prematurely, we forced result clicks to open a new browser tab. Through this questionnaire we collected explicit satisfaction labels that we later used as ground truth to train and evaluate the CAS model.

Each user saw the pop-up questionnaire no more than ten times a day and only for 50% of the sessions. The questionnaire was equipped with "mute buttons" that allowed the user to disable the questions for one hour or 24 hours. We assume that this questionnaire, if not shown overly frequently, would not seriously affect the overall user experience. A similar setup was used in [63].

[6] Our code, including modifications to `EMU.js`, is available at `https://github.com/varepsilon/cas-eval`.

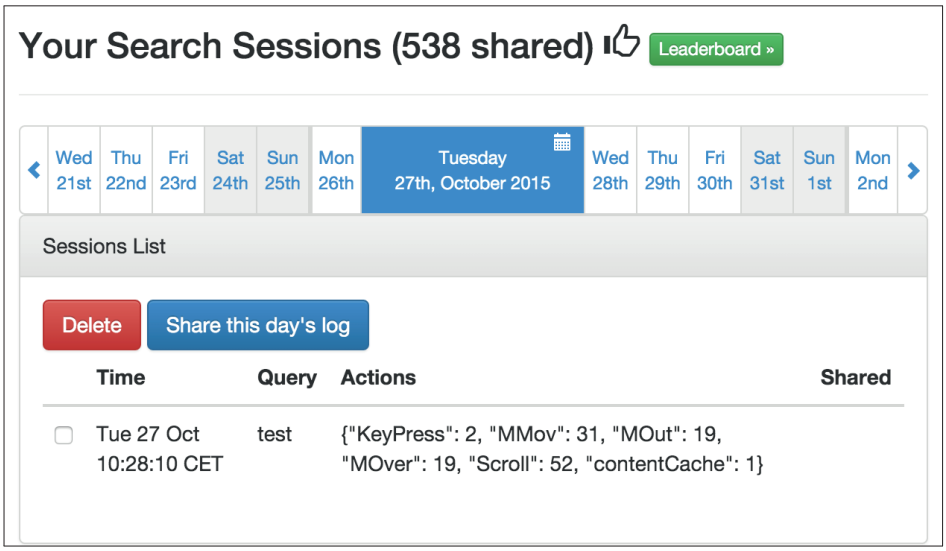


Figure 7.6: Log management interface for experiment participants.

Table 7.3: Data collected with the search proxy.

# of participants	12
# of shared sessions (queries)	2,334
# of shared sessions with satisfaction feedback	243

The parameters of the dataset are summarized in Table 7.3.

7.6.2 Crowdsourcing Data Collection

As a second stage of our experiment we asked crowdsourcing raters ("workers") to assign (D) and (R) labels (see Section 7.5) by showing them SERP items or corresponding web documents and asking the following questions:

(D) Does the text above answer the question Q?

(R) Does the document that you see after clicking the link contain an answer to the question Q?

For the first question we showed only the part of the SERP corresponding to a single SERP item and no clickable links. For the second one we only showed a link and required the workers to click it. Moreover, the above two tasks were run separately so the chances of raters confusing the two tasks were quite low. When comparing the most common (D) and (R) labels assigned for each document, they show Pearson correlation values of 0.085 and Spearman correlation values of only 0.094, which proves that they are quite different.

Originally, a third question was also included to collect *attractiveness* labels ("(A)-ratings") to be used instead of (R) relevance in (7.9). It ran as follows: "Above is a

Table 7.4: Data collected via crowdsourcing. We sent for rating all the sessions with satisfaction feedback (Table 7.3) apart from non-English queries.

	(D)	(R)
# of workers	1,822	951
# of ratings	23,000	22,056
# of snippets/documents rated	2,180	2,180

summary extracted from a bigger document. Do you think examining the full document will be useful to answer the question Q?" However, this proved to be a very subjective question, and attractiveness labels collected this way were less useful as click predictor compared to relevance labels (R). To be precise, the average (A)-rating for the clicked results was 0.82, while it was 0.84 for non-clicked (0.02 standard deviation for both). For the (R)-ratings the corresponding numbers were as follows: 2.29 (standard deviation of 0.29) for clicked and 2.19 (standard deviation of 0.31) for non-clicked. That proves that (R) serves better as a click predictor.

From preliminary runs of the crowdsourcing experiment we learned that the crowd workers rarely pay attention to the detailed instructions of a task, so we decided against using terms like "query" (we used *question* instead) or "snippet" (we referred to it as *text* or *summary*). After several iterations of improving the task we also decided to ask the raters to provide justifications for their answers. We later used this as an additional signal to filter out spammers (see Section 7.9), but it can also be used to understand more about the complexity of individual questions or the task as a whole [10]. One application for the data collected in this way we already saw when we discussed the satisfaction model in Section 7.4.3. Another analysis that we ran was to identify potential good abandonments, i.e., queries that may be answered directly on a SERP [104]. We found out that, even though the raters often disagree with themselves,[7] the queries that were marked as potential good abandonments most often by the raters, were all labeled as such in an independent rating.

An example of the task interface is shown in Figure 7.7. We used the CrowdFlower platform, which was the only platform we could use at the time due to the regional limitations of the other platforms mentioned in Section 2.4. Workers were paid \$0.02 per task to keep the hourly pay above \$1, above the minimum wage in the author's home country[8] and a psychological threshold for the raters to treat it as a fair pay.[9]

The key parameters of the dataset that we collected in this manner are summarized in Table 7.4.[10] After removing ratings coming from spammers (see Appendix 7.9) and sessions that are labeled as something other than "I am satisfied" or "I am **not** satisfied"

[7] Approximately 30% of the raters said that a query is both a potential good and bad abandonment when a slightly different wording was used (or indicated that a potential bad abandonment query has an answer on a SERP).

[8] ₽7,500 per month as of 01.01.2017 (`http://www.consultant.ru/document/cons_doc_LAW_198850/`), roughly equivalent to ₽50 or \$0.80 per hour after accounting for state holidays and vacation days.

[9] The workers were shown an optional survey at the end of the task where they rated "Pay" from 3.2 to 3.5 (out of 5).

[10] The anonymized version of the dataset can be obtained at `http://ilps.science.uva.nl/resources/cas-eval`.

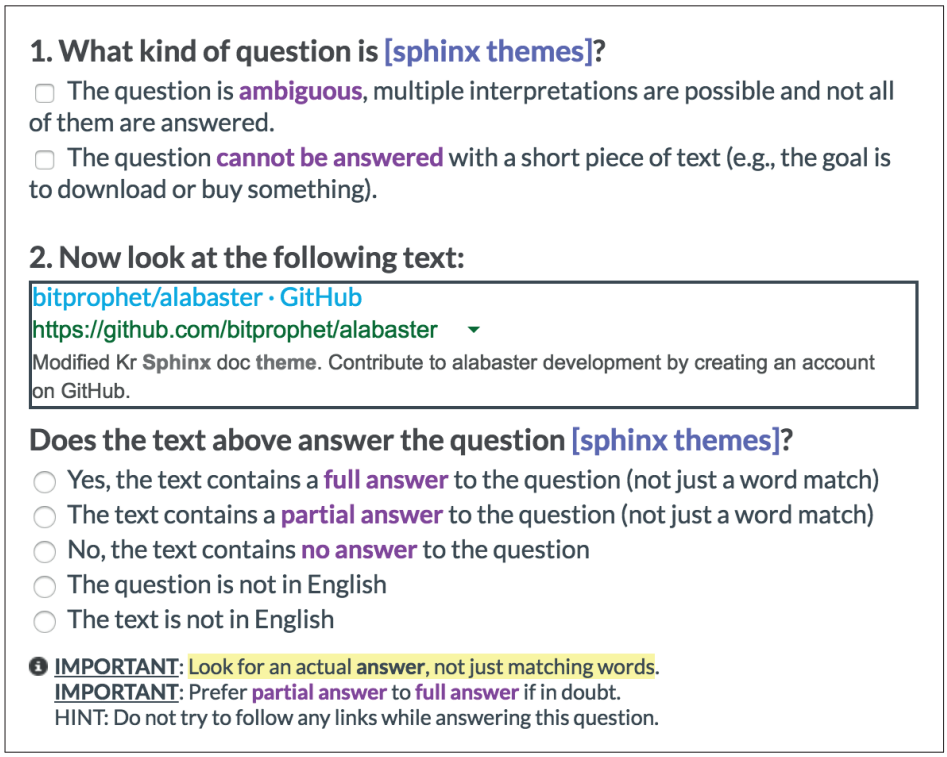


Figure 7.7: Crowdsourcing task for assigning direct relevance label (D) plus some additional questions.

(see Figure 7.5) we are left with 199 query sessions. Of those, 74% were marked as satisfactory; 12% (24 items) of the SERPs are heterogeneous, meaning that they have something other than "ten blue links." For these 199 queries we have 1,739 rated results. If an item does not have a rating, we assume the lowest rating 0, although more advanced approaches exist [11, 20].

7.6.3 Baseline models/metrics

To evaluate our CAS model, we implement the following baseline models:

- the *user browsing model* (UBM) (Section 3.1.5) that was shown to yield a metric that is well correlated with user signals (see Chapter 6);
- the *position-based model* (PBM) (Section 3.1.2), a robust model with fewer parameters than UBM, yet a powerful one (see Chapter 5);
- the *random* model that predicts click and satisfaction with fixed probabilities.[11]

[11] Note that this model is different from the *random click model* (RCM) introduced in Section 3.1.1 in that it learns not only the probability of click, but also the probability of satisfaction.

Apart from these, we also included the following metrics:

- the *discounted cumulative gain* (DCG) metric (Section 3.2.4); and
- the *utility-based UBM metric* (uUBM), the metric that showed the best results in Chapter 6. It is similar to the above UBM model, but parameters are trained not on our dataset (Table 7.4), but on a much bigger dataset (see Chapter 6).

This way we include both non-model-based (DCG) and model-based metrics (the rest), but also locally trained models (UBM, PBM) as well as the uUBM model trained on a different dataset.

To test stability of our results we employ 5-fold cross-validation that we restart 5 times, each time reshuffling the data, see Algorithm 7.1. Thus, we have 25 experimental outcomes that we aggregate to assess significance of the results.

Algorithm 7.1 TQ-fold cross-validation.

procedure TQFOLD(dataset D, T repetitions, Q folds)
  $N \leftarrow size(D)$
  for $i \leftarrow 1$ to T **do**
    $D \leftarrow RandomShuffle(D)$
    for $j \leftarrow 1$ to Q **do**
      $D_{\text{test}} \leftarrow D\left[\frac{N}{Q}(i-1)\ldots\frac{N}{Q}i\right]$
      $D_{\text{train}} \leftarrow D \backslash D_{\text{test}}$
      train on D_{train}
      evaluate on D_{test}

7.7 Results

Below we report results on comparing the CAS model and corresponding evaluation metric to other models and metrics, respectively.

7.7.1 Evaluating the CAS Model

We evaluate the CAS model by comparing the log-likelihood values for different events, viz. clicks and satisfaction. We also analyse the contribution of different attention features introduced in Table 7.2.

Likelihood of clicks

First, we would like to know how the CAS model compares to the baseline models in terms of log-likelihood. Figure 7.8 shows the likelihood of *clicks* (Section 5.2) for different models. On top of the CAS model described above, we include three modifications of it:

- **CASnod** is a stripped-down version of CAS that does not use (D) labels;

- **CASnosat** is a version of the CAS model that does not include the satisfaction term (7.8) while optimizing the model; and
- **CASnoreg** is a version of the CAS model that does not use regularization while training.[12]

As we can see from Figure 7.8, the difference between different variants of CAS is minimal in terms of click log-likelihood. UBM and PBM show better log-likelihood values on average, with PBM being more robust. There are two reasons for CAS to underperform here. First, it is trained to optimize the full likelihood, which includes moused results and satisfaction, not just the likelihood of clicks. As we will see later, CAS shows much better likelihood for satisfaction, more than enough to make up for a slight loss in click likelihood. Second, the class of models for examination and attractiveness probabilities we have chosen (logistic regression) may not be flexible enough compared to the arbitrary rating-to-probability mappings used by PBM and UBM. While similar rating-to-probability mappings can be incorporated into CAS as well, it makes the training process much harder and we leave it for future work.

Likelihood of satisfaction

Next, we look into the log-likelihood of the satisfaction predicted by the various models; see Figure 7.9. For the models that do not have a notion of satisfaction (CASnosat, UBM, PBM, uUBM), we use the sigmoid transformation of the utility function, which, in turn, is computed as the expected sum of relevance of clicked results (see [42]). However, all such models turn out to be inferior to the random baseline. This finding supports the idea of collecting satisfaction feedback directly from the user instead of relying on an ad-hoc interpretation of utility which may be quite different from the user's perception of satisfaction.

By comparing the results for CAS vs. CASnoreg in Figure 7.9 we also see that regularization leads to a more stable satisfaction prediction likelihood, which is, however, lower on average. If we have a large sample of data that is representative of the user population, regularization may as well be omitted. By comparing the performance of CAS vs. CASnod we can also see that the lack of (D) ratings clearly hurts the model's performance as it can no longer explain some utility that is directly gained from the SERP.

Analyzing the attention features

Finally, we look at the features used by the attention model (Table 7.2). If we exclude some of these features we obtain the following simplified versions of the CAS model:

- **CASrank** is the model that only uses the rank to predict attention; this makes the attention model very similar to PBM and the existence of the satisfaction component (7.8) is what makes the biggest difference;
- **CASnogeom** is the model that only uses the rank and the SERP item type information but does not use geometry; and

[12] All other models are trained with L_2-regularization.

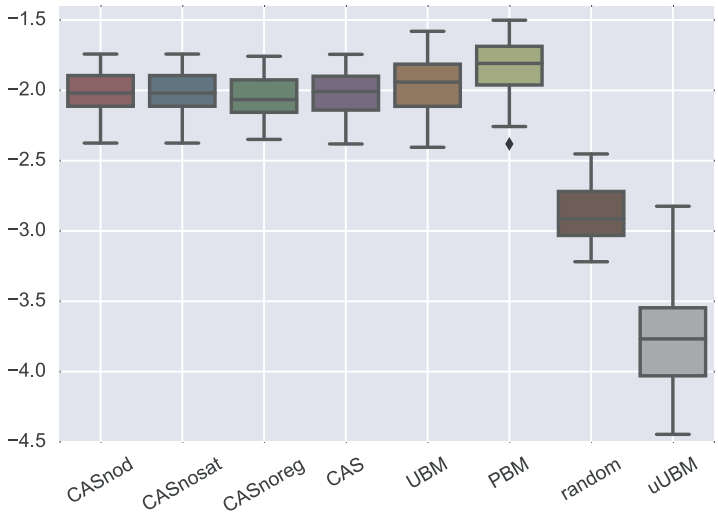


Figure 7.8: Log-likelihood of the click data. Note that uUBM was trained on a different dataset.

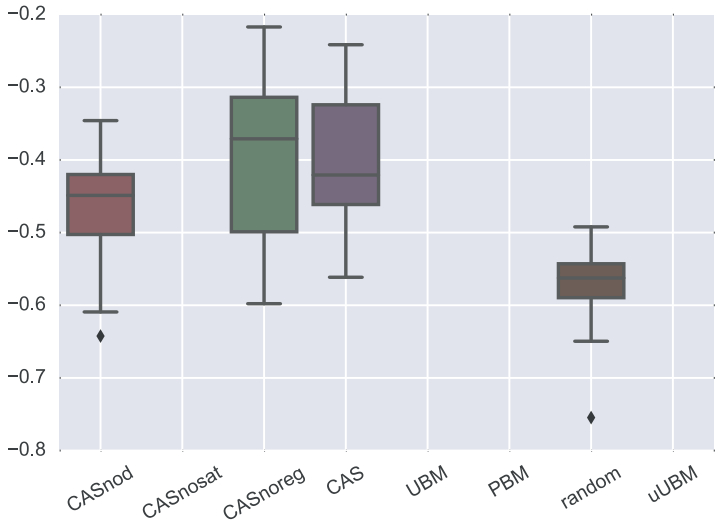
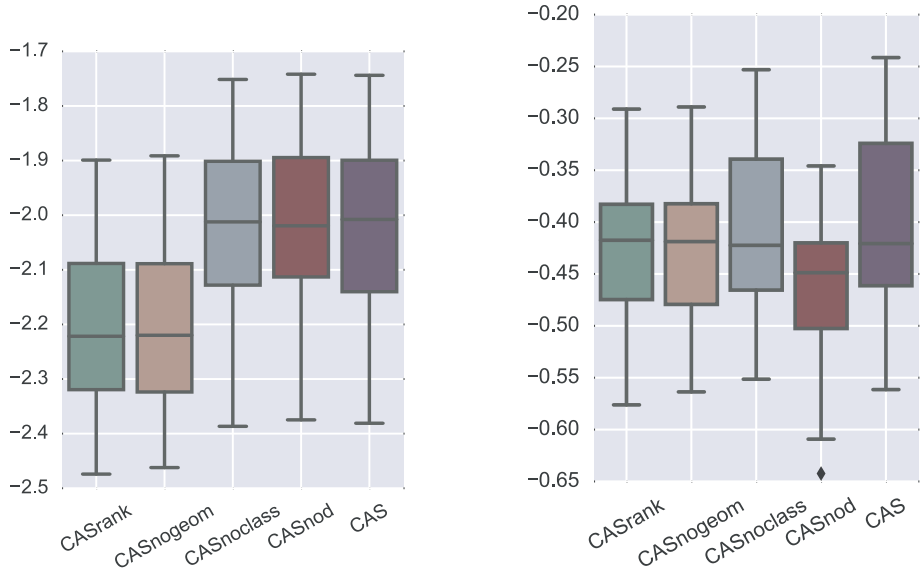


Figure 7.9: Log-likelihood of the satisfaction prediction. Some models here always have log-likelihood below -0.8, hence there are no boxes for them.



(a) Clicks.

(b) Satisfaction.

Figure 7.10: Feature ablation for the attention model: log-likelihood of the click prediction (a) and the satisfaction prediction (b) for vanilla CAS as well as stripped-down versions of it.

- **CASnoclass** is the model that does not use the CSS class features (SERP item type).

We compare these models to the vanilla CAS and CASnod models in terms of log-likelihood of click and satisfaction prediction as we did above for the baseline models.

The results are shown in Figures 7.10(a) and 7.10(b). What we can see from these plots is that excluding (D) labels (CASnod) almost does not affect click prediction accuracy, but it does substantially hurt the satisfaction prediction. This is expected as these labels are only used in the satisfaction formula (7.8). On the other hand, removing geometry features (CASnogeom, CASrank) hurts click prediction the most, while having a less prominent impact on satisfaction prediction. Finally, removing CSS class features (CASnoclass) has a small effect on both click and satisfaction prediction, but much smaller than removing geometry affects click prediction or removing (D) labels affects satisfaction prediction.

In this section we showed that the CAS model predicts clicks slightly worse than the baseline models, albeit at roughly the same level. When it comes to predicting satisfaction events, the baseline models show much lower log-likelihood values, the only comparable performance is shown by the random model, but it performs worse than CAS. In terms of incorporating satisfaction into our models, we demonstrated that it is necessary to do so in order to beat the random baseline on the log-likelihood of satisfaction (CASnosat is always worse than the baseline) and the (D)-labels play an essential role for model

Table 7.5: Correlation between metrics measured by average Pearson's correlation coefficient.

	CASnosat	CASnoreg	CAS	UBM	PBM	DCG	uUBM
CASnod	0.593	0.564	0.633	0.470	0.487	0.546	0.441
CASnosat		0.664	0.715	0.707	0.668	0.735	0.684
CASnoreg			0.974	0.363	0.379	0.417	0.341
CAS				0.377	0.394	0.440	0.360
UBM					0.814	0.972	0.882
PBM						0.906	0.965
DCG							0.943

accuracy: CASnod shows lower log-likelihood than the CAS. This answers our first research question **RQ4.1**.

7.7.2 Evaluating the CAS Metric

Now we evaluate the metric derived from the CAS model and described in Section 7.5. To do this we compute correlations with baseline metrics and with user-reported satisfaction.

Correlation between metrics

Table 7.5 shows the average Pearson correlation between utilities produced by different metrics averaged across folds and repetitions of the cross-validation procedure. As we can see, metrics from the CAS family are less correlated with the baseline metrics than they are with each other. The highest level of correlation with the baseline metrics among the CAS metrics is achieved by CASnosat, the metric that does not explicitly include satisfaction in the user model. This is expected as its model is close to PBM. Another observation from Table 7.5 is that CASnod is also quite different from the baseline metrics, but not as much as CASnoreg and CAS, which, again, shows that including (D) relevance labels (direct snippet relevance) makes the metric quite different.

Correlation with user-reported satisfaction

Figure 7.11 shows the Pearson correlation between the utility induced by one of the models and the binary satisfaction labels reported by the user. As we can see from the plot, the metric induced by the CAS model shows the best Pearson correlation values, despite the fact that it was trained to maximize the full likelihood of the data, not just to predict satisfaction. Correlation is always above zero for metrics based on CAS and CASnoreg, but for the metrics based on CASnod and CASnosat the correlation can be negative, which, again, reinforces the importance of the (D) labels and the explicit satisfaction component in the model. While comparing CAS to the baseline models, we observed that the correlation values for the CAS-based metrics are at least 0.14 higher on average.

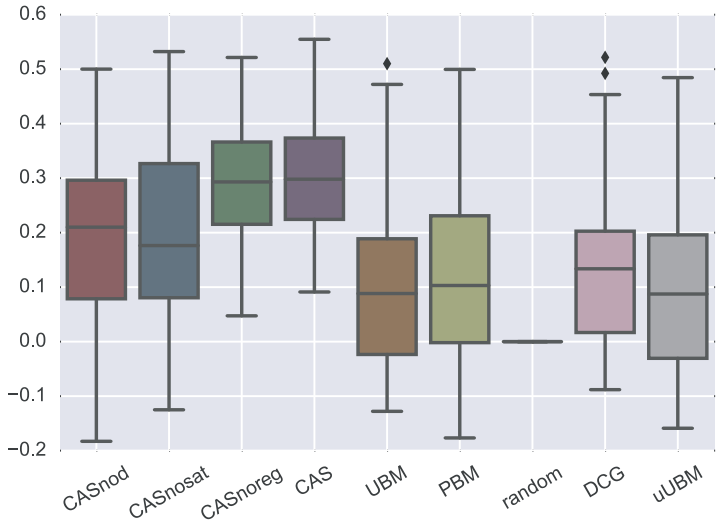


Figure 7.11: Pearson correlation coefficient between different evaluation metrics and the user-reported satisfaction.

Modern SERPs

To prove that the CAS model is especially useful in case of heterogeneous SERPs we perform the following experiment. We take a stratified random split of the dataset into training and testing, where the test set contains $1/24$ of the data and exactly one heterogeneous SERP.[13] We then compute utility of this one SERP using the metric trained on the train set and compare it to the satisfaction label for the corresponding session. This process is repeated 20 times and the Pearson correlation of the utilities and satisfaction labels is computed. Results are reported in Figure 7.12. We see that the CAS and CASnoreg metrics show much higher correlation with the user-reported satisfaction than the other metrics.

Analyzing the attention features

Similar to our analysis in Section 7.7.1 we perform an ablation study, this time to compare vanilla CAS to CASrank, CASnogeom, CASnoclass and CASnod in terms of how well the metrics induced by them are correlated with user-reported satisfaction. The results are shown in Figure 7.13.

As can be see from the plot, removing the class features reduces correlation only a little (CAS vs. CASnoclass). We hypothesize that the reason for this is that in our dataset only 12% of the SERPs have non-trivial SERP items. Removing geometry features (CASnogeom) or both geometry and class features (CASrank) already makes the metric

[13] As we mentioned in Section 7.6.2, our dataset contains 24 such SERPs.

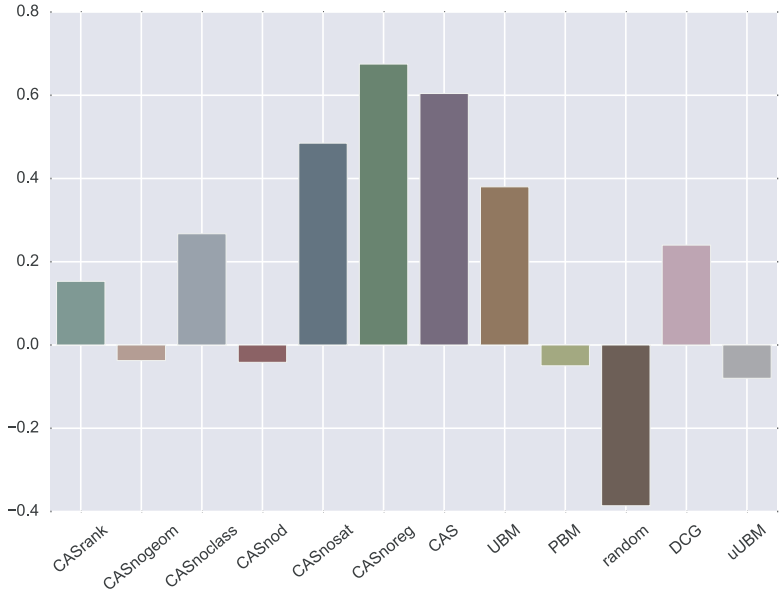


Figure 7.12: Pearson correlation coefficient between utility of heterogeneous SERP and user-reported satisfaction.

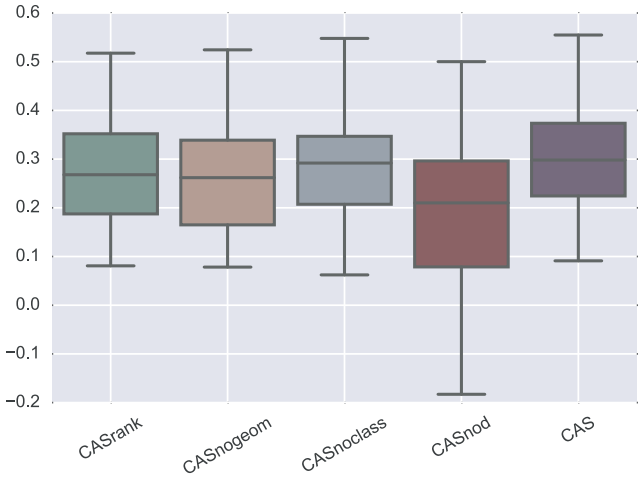


Figure 7.13: Feature ablation for the attention model: Pearson correlation coefficient between different variants of the CAS metric and users' satisfaction.

perform worse, suggesting that these features are crucial for modeling the user's attention on a modern SERP. Finally, the worst performing metric is CASnod which does not use the (D)-labels. The performance drop is much higher than for the models discussed above, which shows that attention features are important for satisfaction prediction, but having (D)-labels brings more to the table. This is consistent with the analysis of the results reported in Figure 7.10(b).

In this section we showed that the metric based on the CAS model differs substantially from the baseline metrics, but less so if the model does not include (D) labels or disregards the satisfaction term altogether. More importantly, the CAS metric is not just different from the baseline metrics, it also shows better correlation with the satisfaction reported by users. So, indeed, incorporating satisfaction yields a new and interesting metric, which answers **RQ4.2**.

7.8 Conclusion

In this chapter we presented a model of user behavior that combines clicks, attractiveness and satisfaction in a joint model, which we call the CAS model. A method to estimate the parameters of the model and a Cranfield-style offline evaluation metric based on this model were proposed. We also described the crowdsourcing setup to collect labels for individual documents.

As was demonstrated, the *model* (cf. **RQ4.1**) conceived in this way can be used as a robust predictor of user satisfaction without sacrificing its ability to predict clicks. We have also shown that decoupling satisfaction from attention and clicks leads to inferior satisfaction prediction without gain in predicting clicks.

In addition, a *metric* was presented (cf. **RQ4.2**) that can be used for offline search system evaluation, an important component of ranking development. The CAS metric with parameters trained from user data consistently shows correlation with satisfaction, unlike traditional metrics. Moreover, the metric is quite different, suggesting that including it into one's evaluation suite may lead to a different view on which version of the ranking system is better.

Future directions. First of all, we would like to acknowledge some limitations of the study presented in this chapter. Our dataset is small compared to the typical datasets used for training click models (cf. Sections 4.4 and 5.4.1) and may be somewhat biased in terms of query distribution since most of the users whose data was used have a computer science background. It would be preferable to collect such data at a bigger scale. One direction for future work would be to train the CAS model on heterogeneous data, where potentially a bigger dataset with clicks and mousing is supplemented by a smaller one with satisfaction labels.

Feature engineering for the attention model also was not comprehensive and was not a goal of this chapter. One may add more saliency features to detect the users' attention or even train separate skimming and reading models [108].

Another challenging part in our setup is the use of crowd workers. It would be interesting to run a study with trained raters and learn how to extrapolate it to the crowd, by adjusting the instructions and filtering the spammers in a more automated fashion than

Table 7.6: Filtered out workers and agreement scores for remaining workers.

label	% of workers removed	% of ratings removed	Cohen's kappa	Krippendorff's alpha
(D)	32%	27%	0.339	0.144
(R)	41%	29%	0.348	0.117

we have used [95]. There is also a noticeable difference between raters and the users. For example, Liu et al. [109] claim that the raters pay more attention to the effort required to complete a task, while the users care more about utility. Also, the ratings assigned by the owners of the query are different from the ones assigned by other people [31].

Mobile search evaluation [78, 86] is another facet of future work. As we mentioned in the introduction, navigating away from a SERP is more expensive there, so users tend to gain utility directly from the SERP and the search engines add more ways to aid this behavior. It would be interesting to see how we can leverage additional attention signals to adapt the CAS model for mobile settings.

While the experiments presented in this chapter have limitations, we view them as a motivation to move away from the ten blue links approach and adopt an evaluation metric that uses rich features and relevance signals beyond traditional document relevance. We also call for releasing a dataset that would allow for a more comprehensive evaluation than currently provided by TREC-style evaluation setups.

In the current chapter and the one before we talked about offline quality evaluation based on expert ratings. Next, in Chapter 8 we are going to look at a different type of evaluation, namely online interleaving evaluation, and show how it can be adjusted for modern SERPs.

7.9 Appendix: Filtering Spammers

To identify spammers, we use free-text fields where the raters were asked to copy text from the snippet or full document to support their relevance ratings. If the text was not copied from the snippet (in case of direct snippet relevance) or contained gibberish words, we add this worker to the list of suspicious workers. After each batch of tasks sent for ratings was finished, we manually review lowest scoring workers according to those metrics and ban them from the future tasks.[14] We also ignore workers with fewer than three ratings following [10]. In total, we ignore ratings coming from 698 workers out of 2,185, which corresponds to 27% of direct snippet relevance ratings (D) and 29% of relevance ratings (R), see Table 7.6.

To measure worker disagreement we report average Cohen's kappa [58] as well as Krippendorff's alpha [103]. The numbers are reported in Table 7.6. As we can see, the agreement numbers are rather low, which shows that there is still a big variation of opinions.

[14] Manual examination of the worker's ratings before banning is enforced by the crowdsourcing platform that we used.

Fortunately, our model is able to accommodate this by taking the histogram of ratings and not just a single number coming from averaging or majority vote; see (7.5), (7.6) and (7.9).

We also experimented with worker-worker and worker-task disagreement scores [10], the method of cleaning the data where workers that disagree with too many other workers on either global or per-item level are removed from the data. We explored different thresholds on disagreement scores and managed to improve overall agreement measured by Cohen's kappa and Krippendorff's alpha, which was expected, but it did not improve the final results (Section 7.7). Presumably, the reason for that was that there were always enough careless workers that consistently gave wrong answers and showed good agreement with each other, and the method based on worker disagreement was not able to catch them. Moreover, some disagreement is natural in such a subjective task; reducing it does not necessarily improve quality.

Finding a good balance between data quantity and data quality is a topic for a different discussion and is outside the scope of this chapter. Changing the settings for spammer filtering to very aggressive (removing all workers that made at least one mistake thus filtering out over 75% of the ratings) or to very permitting (no spammer filtering) gives rise to models with inferior performance both in terms of likelihood of clicks/satisfaction and the correlation of utility and user-reported satisfaction.

8

Online Evaluation of Modern Search: Interleaving

Unlike the previous two chapters, here we focus on online evaluation where a system's quality is inferred from real users' interactions with a live search system. As we said in the introduction, modern SERPs are intrinsically heterogeneous and thus create challenges for us when we try to interpret user behavior. An additional problem arises for so-called *interleaving* methods—by mixing results from two different SERPs one can easily break the desired web layout in which vertical documents are grouped together, and hence affect the user experience, potentially in a negative way.

In this chapter we conduct an analysis of different interleaving methods as applied to modern heterogeneous SERPs. In addition to conventional interleaving methods, we propose two vertical-aware methods: one derived from the widely used team-draft interleaving method by adjusting it in such a way that it respects vertical document groupings, and another based on the recently introduced optimized interleaving framework. We show that our proposed methods are better at preserving the user experience than existing interleaving methods while still performing well as a tool for comparing ranking systems. For evaluating our proposed vertical-aware interleaving methods we use real world click data as well as simulated clicks and simulated ranking systems.

Our main research questions are as follows:

RQ5.1 **Influence on the user experience.** What effect do different interleaving methods—both conventional and newly introduced vertical-aware—have on the user experience in the case of complex SERPs? Do any of these methods run the risk of degrading the quality of the results or altering the user experience?

RQ5.2 **Correctness and sensitivity.** Do different interleaving methods always draw correct conclusions about the better system? How fast, in terms of the number of impressions and amount of feedback needed, can they detect that one aggregated search system is to be preferred over another?

RQ5.3 **Unbiasedness.** Do the interleaving methods that we consider provide a fair and unbiased comparison, or do some of them erroneously infer a preference for one aggregated search system over another in situations where implicit feedback is provided by a randomly clicking user?

In order to answer **RQ5.1** we consider different pairs of SERPs and analyse the effect on the user experience imposed by the interleaving methods that we consider. We analyze a scenario where the aggregated result pages are allowed to have multiple blocks that originate from different verticals. In addition to evaluating visual changes, we add an evaluation of the quality of the interleaved list as captured by both click metrics and offline rater-based metrics. When evaluating visual changes we aim at *preserving* the usual user experience and alter it as little as possible. When we look at offline or online metric-based quality, we want to make sure that we *do not degrade* the user experience.

RQ5.2 concerns the ability of an interleaving method to correctly capture the difference between two rankers by using the minimal amount of implicit user feedback. We analyze the ability to notice strong differences between rankers (formulated in terms of Pareto dominance), as well as subtle differences reported by offline and online quality metrics (including those specific to aggregated search).

Finally, to answer **RQ5.3** we check that none of the evaluation methods infer statistically significant preferences when we assume a randomly clicking user.

8.1 Introduction

There is a general trend to aggregate search results from different verticals (e.g., News, Image, Video) and present them in one search result page together with "general web" or "organic web" results. This new search paradigm is called *aggregated search* or *federated search* and has been adopted by all major commercial search engines. As much as one third of all queries have an intent that can be answered by a specific vertical, whereas 10% to 30% of the queries require at least two different types of vertical. This was first pointed out in a study by Arguello et al. [5], where human editors found at least two relevant verticals for 30% of the queries in a random sample of Yahoo! Search[1] queries. More conservative estimations from other search engines were later reported by Chen et al. [28] and Styskin [152]. With respect to presentation style, it has become standard to group the results coming from the same vertical and present them as one coherent block. As was shown by Dumais et al. [68], presenting results in a grouped manner simplifies browsing result lists and helps users to navigate faster.

As web search engines constantly evolve, their quality needs to be evaluated. *Interleaving* [90, 91] is an evaluation method for comparing the relative quality of two ranking systems. Users are presented with a mixed or *interleaved* list of results from both rankings and the users' clicks are used to determine which system is better. Interleaving methods have proved to be an efficient tool; they allow us to identify the preferred system from user clicks faster than other methods such as A/B-testing [127].

How can interleaving methods be used to measure the quality of aggregated search systems that combine web and vertical results? We want to achieve the efficiency of conventional interleaving methods while preserving the conventional aggregated search user experience, in which vertical results are organized in blocks. In this chapter we propose several extensions of interleaving methods to handle complex heterogeneous search engine result page comprising results from different verticals.

[1] `https://search.yahoo.com`

The rest of the chapter is organized as follows. We discuss related work in Section 8.2. In Section 8.3 we introduce conventional interleaving methods and their limitations as applied to modern search result pages. In Section 8.4 we present two *vertical-aware* interleaving methods capable of dealing with not just one but many vertical blocks of different types. Section 8.5 describes the experimental setups that we use and discuss their strengths and weaknesses. In Sections 8.6, 8.7 and 8.8 we address the research questions formulated above. We conclude in Section 8.9.

8.2 Related Work

Two lines of research relate to this work. One focuses on evaluating aggregated SERPs using different evaluation paradigms. The other explores utilizing interleaving methods for evaluating search rankings.

Vertical Search

Vertical or *aggregated search* deals with presenting results from different verticals in one unified interface. We discuss the foundations of it in Section 2.2.

Evaluation of aggregated search is a complex and challenging problem as there is a variety of compounding factors. Four key components of aggregated search have the main influence of the user's experience: *vertical selection* (VS), *vertical diversity* (VD), *item selection* (IS), *result presentation* (RP) (cf. Section 5.3). Vertical selection and diversity deal with deciding which set of diverse verticals are implicitly intended by a query. Item selection deals with selecting a subset of items from each vertical to present on the aggregated page. Result presentation deals with organizing and embedding the various types of result on the result page.

There is a growing body of work on evaluating aggregated search. Relatively early work considered VS to be the main criterion of evaluation and much of the research [5, 172] aimed to measure the quality of the set of selected verticals, compared with an annotated set obtained by collecting manual labels from assessors underlying different assumptions [173, 175].

In later work, to evaluate both VS and RP, Arguello et al. [8] proposed to use pairwise preference evaluation to rank the vertical blocks for a given query. Targeting a similar objective, Ponnuswami et al. [122] described the process of manual assessments of both vertical and organic documents, and proposed a click-based assessment of vertical block placement similar to Joachims [90]. However, they neither discussed the combined effect of these two ranking aspects nor suggested a way to compare vertical documents inside one block to each other (IS). Most of these evaluation methods assumed that the vertical block was atomic, i.e., it was considered to have no internal structure. However, this is not always the case. For example, a news block usually contains a list of headlines. Each of them can be judged separately and compared to organic web results.

Recently, Zhou et al. [171] followed the Cranfield paradigm (Section 2.4) and proposed an evaluation framework for measuring aggregated search quality by modeling all, or a subset, of the four aggregated search key components discussed above (VS, VD, IS and RP). The main differences between these proposed metrics are the way they model each

factor and the way they combine them. A list of diversity metrics such as α-NDCG [52] was also adapted to evaluate aggregated search. By meta-evaluating those metrics on their *reliability* and *intuitiveness*, Zhou et al. [174] concluded that the AS_{RBP} [171] was the preferred metric. Similar to [171], we aim to capture the four key components of aggregated search together when we conduct our evaluations. To answer **RQ5.1**, we also utilize a set of offline metrics, including AS_{RBP}, to quantify the quality of the results in order to verify that our proposed approaches do not degrade the user experience.

Interleaving

We build vertical-aware interleaving methods on top of TDI and on top of OI that we describe in Section 2.5. Below we mention the most relevant related work in the area of interleaving and evaluation thereof and put our research in context.

Most interleaving methods presented until now, were evaluated in terms of *sensitivity* and *correctness*, which corresponds to our **RQ5.2**. Radlinski and Craswell [123] showed that interleaving is more accurate and more sensitive than standard offline metrics such as NDCG or ERR introduced in Section 3.2. Radlinski and Craswell [124] applied the same evaluation method to show that OI is more sensitive than TDI if the right credit function is used. They also analyzed typical biases of different interleaving methods (related to our **RQ5.3**).

Another meta-evaluation method was proposed by He et al. [79] where different interleaving methods were compared in terms of their effect on the user experience (related to our **RQ5.1**). Unfortunately, they did not report any difference between interleaving methods nor did they compare the quality of the interleaved system to the original A and B systems being interleaved. We substantially extend this meta-evaluation method in our Section 8.6 (**RQ5.1**).

A comprehensive study with a systematic analysis of interleaving methods and their meta-evaluation was suggested by Hofmann et al. [82]. The main questions we aim to answer are related to, but different from, the notions of fidelity, soundness and efficiency studied by Hofmann et al. [82]. First of all, we put more focus on the *user experience* aspect (**RQ5.1**), which was not analyzed in [82]. Second, our **RQ5.2** unites two closely related questions of *sensitivity* and *correctness*, which in [82] were split across definitions of efficiency and fidelity (their Definitions 4.4 and 4.2(2) respectively). And finally, we separate the question of *unbiasedness* which in [82] was bundled with correctness (their Definition 4.2(1)). In addition, we do not study the notion of soundness introduced by Hofmann et al. [82] since it is trivially satisfied for the interleaving methods we use.

Our work differs in important ways from the work just discussed. In contrast to previous work on evaluating aggregated search, we perform online evaluations based on interleaving methods that allow us to evaluate aggregated system as a whole and to do so efficiently. Our proposed approach only needs naturally occurring user interactions, which makes it cheap and useful in an online environment. In contrast to previous work on interleaving, we propose a new interleaving method that preserves the user experience on complex aggregated SERPs while maintaining a high degree of sensitivity and preserving unbiasedness.

8.3 Conventional Interleaving

To design a new interleaving method one can follow one of two possible ways. The first way, taken by the majority of the interleaving methods [80, 91, 127], assumes that the designer of the interleaving method specifies the probability distribution over possible interleaved lists. This probability distribution was made explicit in probabilistic interleaving [80] and left implicit, although easily reconstructible, in balanced interleaving [91] and team-draft interleaving [127].[2] Another approach introduced by Radlinski and Craswell [124] specifies an interleaving method as an optimization problem that allows us to explicitly tune the method for better sensitivity.

According to Joachims [91], a good interleaving method should be blind to the user and have a low usability impact. This is particularly important for vertical search, since, as we will show below, conventional interleaving methods do not fully satisfy these conditions.

We will use two conventional interleaving methods as baseline methods: *team-draft interleaving* (TDI) and *optimized interleaving* (OI). The former is widely used in commercial settings [42, 123], while the latter has good flexibility which enables us to naturally extend it to the aggregated search setting. The same two interleaving methods were also extended in [138] to allow for comparisons of multiple rankers at once.

8.3.1 Team-draft Interleaving (TDI)

Conventional TDI (Algorithm 8.1) allows rankings A and B to contribute documents turn by turn and keeps track of which side contributes which document ($Team_A$ or $Team_B$). By flipping a coin in cases when both teams have the same size (line 4), the algorithm ensures a fair and unbiased team assignment and sufficient variability in the result list L.

Algorithm 8.1 Conventional TDI.

```
1:  function TDI(ranking A, ranking B)
2:      L ← []; Team_A ← ∅; Team_B ← ∅
3:      while |L| < N do
4:          if |Team_A| < |Team_B| + RANDOMINTEGER(0, 1) then
5:              k ← min{i | A[i] ∉ L}
6:              Team_A ← Team_A + A[k]              ▷ add A[k] to Team_A
7:              L ← L + A[k]                        ▷ append A[k] to the list L
8:          else
9:              k ← min{i | B[i] ∉ L}
10:             Team_B ← Team_B + B[k]              ▷ add B[k] to Team_B
11:             L ← L + B[k]                        ▷ append A[k] to the list L
12:     return L
```

As we can see in Algorithm 8.1, there is nothing that prevents result list L from mixing up documents from different verticals with web documents. Even when we have only one vertical block of the same type in A and B, the algorithm may give rise to two blocks in

[2] The distribution is uniform amongst allowed rankings in these two methods.

the interleaved list and therefore affect the user experience. For instance, Figure 8.1 shows a schematic representation of two ranked lists A, B and one of the possible interleaved lists. In this example, vertical documents d_3, d_4 and d_5 are not grouped in the result list L, which deviates from the intended aggregated search user experience.

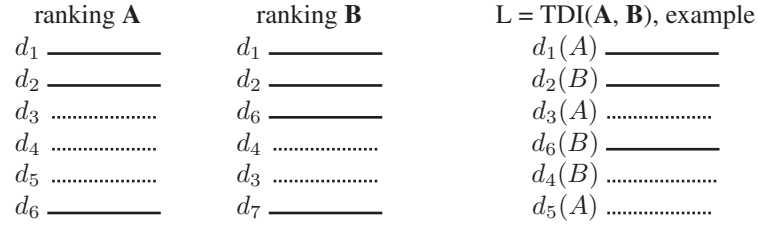


Figure 8.1: Two rankings with a vertical block and one of the possible interleaved lists (TDI). Vertical documents are shown as dotted lines.

8.3.2 Optimized Interleaving (OI)

Radlinski and Craswell [124] proposed to formalize the "low usability impact" constraint (our **RQ5.1**) using a *prefix condition* (called c' there):

$$\forall k \; \exists i, j \text{ such that } L^k = A^i \cup B^j, \tag{8.1}$$

where A^i is the set of top-i documents returned by system A, B^j is the set of top-j documents returned by system B, and L^k is the set of top-k documents of the interleaved ranking L.[3]

When we fix a prefix condition, all we need to do is assign probabilities to all the possible interleaved lists L that conform to this condition (we denote the set of possible interleaved lists as $\mathcal{L}$). Radlinski and Craswell [124] suggested choosing this probability distribution in order to maximize a method's sensitivity while preserving its unbiasedness. We also need to specify a credit function $\delta(d)$ that assigns a particular credit to document d depending on its rank in rankings A and B. If we assume for now that the credit function is fixed, we can write down the following optimization problem:[4]

$$p_i \in [0, 1] \tag{8.2}$$

$$\sum_{i=1}^{|\mathcal{L}|} p_i = 1 \tag{8.3}$$

$$\forall k \in \{1, \ldots, N\} : \sum_{i=1}^{|\mathcal{L}|} p_i \delta_k(L_i) = 0 \tag{8.4}$$

$$\sum_{i=1}^{|\mathcal{L}|} p_i s(L_i) \rightarrow \max_{\{p_i\}}, \tag{8.5}$$

[3] Note that interleaved lists produced by TDI also satisfy the prefix condition (8.1).

[4] Radlinski and Craswell [124] used a different formalization of unbiasedness (Section 3 there), which is, however, mathematically equivalent to our condition (8.4).

where s is an estimated sensitivity function defined as follows:

$$\begin{aligned} s(L) &= -\frac{1-w_T}{w_A+w_B}\left(w_A \log w_A + w_B \log w_B - (w_A+w_B)\log(w_A+w_B)\right) \\ w_A &= \sum_{k:\ \delta_k>0} f(k) \\ w_B &= \sum_{k:\ \delta_k<0} f(k) \\ w_T &= \sum_{k:\ \delta_k=0} f(k) \\ f(k) &= \frac{1/k}{\sum_{m=1}^{N} 1/m}. \end{aligned}$$

We use the same constraints as [124] for the credit function. Given a document d, let $\delta(d)$ denote the credit assigned to system A for this document.[5] Thus, when $\delta(d)$ is positive then A receives credit, if it is negative then B receives credit. The credit function should satisfy:

$$\mathrm{rank}(d,A) < \mathrm{rank}(d,B) \quad \Leftrightarrow \quad \delta(d) > 0 \tag{8.6}$$

$$\mathrm{rank}(d,A) > \mathrm{rank}(d,B) \quad \Leftrightarrow \quad \delta(d) < 0, \tag{8.7}$$

where $\mathrm{rank}(d,X)$ is the position of document d in ranking X (we set $\mathrm{rank}(d,X) = +\infty$ if $d \notin X$). In particular, we set δ to linear rank difference which is the most sensitive setting in [124, equation (14)]:

$$\delta_k(L) = \delta(d_k) = \mathrm{rank}(d_k,B) - \mathrm{rank}(d_k,A). \tag{8.8}$$

The interleaving method then proceeds as specified in Algorithm 8.2.

Algorithm 8.2 Conventional OI.

1: **function** OI(ranking A, ranking B, credit function δ_i)
2: $\mathcal{L} \leftarrow \{L \mid L \text{ satisfies (8.1)}\}$
3: Find a distribution $\{p_L \mid L \in \mathcal{L}\}$ that conforms to the constraints (8.2), (8.3), (8.4) and maximizes the average sensitivity (8.5).
4: Sample L from $\mathcal{L}$ according to the probability distribution p_L
5: **return** L

If we are to add a constraint that discards all the rankings L that split vertical documents of the same type, we will end up with a very biased list of rankings $\mathcal{L}$. Consider the example provided in Figure 8.2. In this example each ranking A and B has exactly one vertical block, say, News, which consists of different documents and resides on positions 4 and 5 in both rankings A and B. We also assume that organic rankings are sufficiently different in A and B. We can easily see that the prefix condition (8.1) does not allow

[5] For simplicity we use the same names for rankers (ranking systems) and the ranking lists they generate. It should be clear from the context what we refer to.

system B to contribute its vertical documents until it has contributed all the documents before the block. The same holds for system A. This means that if we want both systems to contribute to the resulting vertical block of L, we are forced to place the block lower in the ranking, because we first need to show the union of before-block documents from A and B. This implies that, in Figure 8.2, we need to present d_1, d_2, d_3, d_6, d_7 before the vertical block whenever we want both d_4 and d_8 to be present in the resulting ranking. But if we do so, we push the vertical block to the bottom of the result page and hence influence the user experience (**RQ5.1**). We also risk having a bigger vertical block than in either ranking A or ranking B for the same reason.

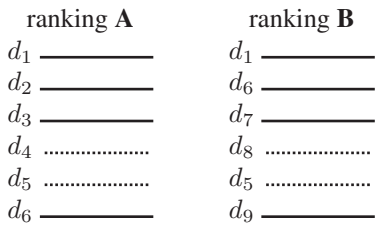


Figure 8.2: Two rankings with a vertical block at the same position.

8.4 Vertical-aware Interleaving

In this section we discuss vertical-aware extensions of the two interleaving methods that we study: TDI and OI. We modify these interleaving methods such that the vertical documents of the same type are guaranteed to be grouped.

8.4.1 Vertical-aware Team-draft Interleaving (VA-TDI)

We describe an algorithm that may be viewed as a generalization of the conventional TDI method by Radlinski et al. [127]. The intuition is to start with the standard TDI method and then alter it to meet the following requirements:

- R1 Both of the ranked lists being interleaved should contribute to the vertical block placement size and position in the interleaved list;
- R2 Both ranked lists should contribute vertical and organic web documents;
- R3 Team assignment should be "fair"; and
- R4 The resulting interleaved list should not degrade the user experience.

We propose a method called *vertical-aware team-draft interleaving* (VA-TDI) (Algorithm 8.3) that generalizes Algorithm 1 from [40] to the case of multiple vertical blocks. The main idea is to enforce the grouping of vertical documents. Therefore, our algorithm proceeds like the conventional TDI method until it hits the first vertical document. After that it interleaves only vertical documents of this type (line 29) until the block has been formed (line 18), i.e., there are no vertical documents left or the desired block size is reached.[6] After that, the algorithm continues as usual, but ignores the vertical documents

[6] We pick the desired block size beforehand (line 7) to ensure requirement R1 is met.

Algorithm 8.3 *Vertical-aware team-draft interleaving* (VA-TDI).

function VATDI(ranking A, ranking B)
  $vLeft \leftarrow \varnothing$ ▷ verticals left (not yet present in L)
  for every vertical type t present in either A or B **do**
    $A_t \leftarrow \{d \in A \mid d \text{ is a vertical doc of type } t\}$
    $B_t \leftarrow \{d \in B \mid d \text{ is a vertical doc of type } t\}$
    $Size_t^A \leftarrow |A_t|$; $Size_t^B \leftarrow |B_t|$
    $Size_t^L \leftarrow$ SAMPLESMOOTHLY$(Size_t^A, Size_t^B)$
    if $Size_t^L \neq 0$ **then**
      $vLeft \leftarrow vLeft \cup \{t\}$
  $L \leftarrow []$
  $Team_A \leftarrow \varnothing$; $Team_B \leftarrow \varnothing$
  $t \leftarrow$ NULL ▷ current vertical type
  while $|L| < N$ **do**
    if $|Team_A| < |Team_B| +$ RANDOMINTEGER$(0, 1)$ **then**
      ADDNEXTDOCFROM$(A, t, vLeft)$
    else
      ADDNEXTDOCFROM$(B, t, vLeft)$
    if $t \neq$ NULL and $|\{d \in L \mid d \text{ is a vertical doc of type } t\}| = Size_t^L$ **then**
      $vLeft \leftarrow vLeft \backslash \{t\}$ ▷ this vertical is completed in L
      $t \leftarrow$ NULL
  return L

function SAMPLESMOOTHLY(integer a, integer b)
  if $a > b$ **then**
    SWAP(a, b)
  Sample r randomly from $[a-1, b+1]$ where all integers from $[a, b]$ have equal probability p; $(a-1)$ and $(b+1)$, each has probability $\frac{p}{2}$
  return r

procedure ADDNEXTDOCFROM(ranking X, current vertical type t, $vLeft$)
  ▷ X is either A or B
  if $t \neq$ NULL **then**
    $X_{left} \leftarrow \{i \mid X[i] \in X_t \backslash L\}$
    if $X_{left} = \varnothing$ **then**
      raise exception ▷ interleaved list is rejected, start a new one
  else
    $X_{left} \leftarrow \{i \mid X[i] \in X \backslash L$ and ($X[i]$ is a Web doc or $\exists t' \in vLeft$ such that $X[i] \in A_{t'}$)$\}$
  $k \leftarrow \min X_{left}$ ▷ select the document with the min rank
  $Team_X \leftarrow Team_X \cup \{X[k]\}$ ▷ add the document to the team
  if $X[k]$ is a vertical doc of type t' **then**
    $t \leftarrow t'$ ▷ change the current vertical type
  $L \leftarrow L + X[k]$ ▷ append the document to L

whose blocks have already been formed (line 33).

If we look back to our original goals, we see that Algorithm 8.3 explicitly chooses block sizes between those of A and B (with some smoothing in order to do exploration), while the position of the block is contributed implicitly (although both ranked lists can influence it). Requirements R2 and R3 are met automatically due to the TDI procedure that we re-use (after one ranked list wins the coin flip and contributes the document, the other ranked list has to contribute the next document), though we also verify them along with requirement R4 in our experiments.

On the other hand, the algorithm proposed has some weaknesses. First of all, it assumes a rejection sampling procedure: if the selected block size cannot be achieved, we reject the list being built and start a new one with another randomization (line 31). This usually happens when A and B have very different block sizes and we cannot build big enough vertical block as VA-TDI assumes that A and B contribute roughly the same number of vertical documents (line 14). Using synthetic rankings (Section 8.5.3), we identify that rejection happens on average 3.8 times per interleaving function invocation with slightly over hundred in the worst case. This could lead to unwanted delays in building the interleaved SERP and may potentially affect the user if the search is run on slow hardware. It is worth noting, however, that the TDI method normally takes a negligible amount of query processing time and even a 100-fold increase should not lead to a visible degradation.[7] Another issue of the VA-TDI method exhibits itself when we consider A and B with different sets of verticals. The method's nature prevents one ranked list from contributing more than two documents in a row, which means that if A has a vertical of type t and size n ($n \geqslant 3$) while B does not have a vertical of type t, the interleaved list will only be able to include one or two top documents from this block. As we will see later, this leads to smaller vertical blocks (Section 8.6.1), which may affect the user experience. This also limits the sensitivity of VA-TDI (due to the fact that it may skip vertical documents)—as we will see in Section 8.7, VA-TDI is unable to break the limit of sensitivity 0.8 even when we consider a more obvious case where A always Pareto dominates B. In the next section we propose another interleaving method, *vertical-aware optimized interleaving* (VA-OI), that does not suffer from these limitations.

8.4.2 Vertical-aware Optimized Interleaving (VA-OI)

In order to extend the optimized interleaving method presented in Section 8.3.2, we extend equation (8.1) and introduce an additional constraint that guarantees that vertical documents of the same type are presented in a grouped manner (8.16).

Consider a list of documents A, containing multiple blocks of vertical documents of different types. Let A_{Web} be the list of organic web results in A, so A stripped of all vertical blocks. For any vertical type $t \in \{\text{Image}, \text{News}, \ldots\}$, let A_t be the list of vertical documents of that type in A. This list may be empty. We assume that all vertical documents of one type occur in A in one group, so all those documents are grouped together in the SERP.

Let A^k_{Web} be the list of the k documents in A_{Web} having the smallest rank values (ranked higher). So for $A_{\text{Web}} = (d_1, \ldots, d_n)$, we have $A^k_{\text{Web}} = (d_1, \ldots, d_k)$. For any

[7] For practical purposes we can also introduce a cut-off and skip interleaving if it takes too much time to build. This would lead, however, to a biased sample and requires a separate analysis.

vertical type t, define A_t^k in a similar way. Note that it is not necessarily so that a document d_i in $A_{\text{Web}} = (d_1, \ldots, d_n)$ is ranked at position i, since there may be vertical documents above d_i, pushing it to a lower rank. Therefore, there may be documents in A^k that are ranked below rank k. For a document d, let $\text{rank}(d, A)$ denote its rank in A, with $\text{rank}(d, A) = +\infty$ if d is not in A. For a set of documents S we define $\text{rank}(S, A) = \min_{d \in S} \text{rank}(d, A)$.

From two lists A and B, we generate a new list L that is a combination of the two. We define similar lists and functions for B and L as above for A. The constraints that should be satisfied by any generated list are:

$$\forall k\ \exists i, j : L^k_{\text{Web}} = A^i_{\text{Web}} \cup B^j_{\text{Web}} \tag{8.9}$$

$$\forall t, k\ \exists i, j : L^k_t = A^i_t \cup B^j_t \tag{8.10}$$

$$\forall t \text{ such that } L_t \neq \varnothing : \min(|A_t|, |B_t|) \leqslant |L_t| \leqslant \max(|A_t|, |B_t|) \tag{8.11}$$

$$\forall t \text{ such that } L_t \neq \varnothing : \text{rank}(L_t, L) \geqslant \min(\text{rank}(A_t, A), \text{rank}(B_t, B)) \tag{8.12}$$

$$\forall t \text{ such that } L_t \neq \varnothing : \text{rank}(L_t, L) \leqslant \max(\text{rank}(A_t, A), \text{rank}(B_t, B)) \tag{8.13}$$

$$|\{t \mid L_t \neq \varnothing\}| \geqslant \min(|\{t \mid A_t \neq \varnothing\}|, |\{t \mid B_t \neq \varnothing\}|) \tag{8.14}$$

$$|\{t \mid L_t \neq \varnothing\}| \leqslant \max(|\{t \mid A_t \neq \varnothing\}|, |\{t \mid B_t \neq \varnothing\}|) \tag{8.15}$$

$$\forall t\ \exists m :\ \forall k \in [m, m + |L_t|] :\ \exists d \in L_t : \text{rank}(d, L) = k. \tag{8.16}$$

Constraint (8.9) is the prefix constraint introduced by Radlinski and Craswell [124]. It requires any prefix of L_{Web} to be a combination of the top i and top j documents of A_{Web} and B_{Web}. This essentially means that the new list could be constructed by repeatedly taking the top document that is not used yet from either A_{Web} or B_{Web}. Constraint (8.10) is an adaptation of the prefix constraint for verticals. It requires that the prefix of any vertical block L_t is a combination of top documents of the lists of verticals of the same type in A and B. Constraint (8.11) requires the size of each vertical block to be between the sizes of the vertical blocks of the same type in A and in B. Constraints (8.12)–(8.13) control the position of each vertical block. The rank of the vertical block should be between the ranks of the blocks of the same type in A and B.

We need additional constraints to control the number of vertical blocks in the list L. If we only had constraints (8.9)–(8.13), it might be the case that there are three vertical blocks in A, three vertical blocks of different types in B, and six vertical blocks in the result list L. This would change the user experience a lot. Therefore, constraints (8.14) and (8.15) limit the number of vertical blocks in L to be between the number of vertical blocks in A and the number of vertical blocks in B.

Finally, constraint (8.16) ensures that ranks of the vertical documents of the same vertical type t are adjacent numbers, i.e., there are no gaps in the ranks.

The interleaving method then proceeds as described in Algorithm 8.4.

With these constraints, we require all aspects of the interleaved list (document placement in the regular result list, document placement inside the vertical block, size of the vertical block, position of the vertical blocks and number of vertical blocks) to be somewhere between A and B. This allows for exploration of all possible rankings that can be considered a combination of A and B.

It is worth noting that our constraints (8.9)–(8.16) are more limiting than the original constraints in the conventional OI method. As a consequence, the problem of forming an

Algorithm 8.4 *Vertical-aware optimized interleaving* (VA-OI).

1: **function** VAOI(ranking A, ranking B, credit function δ_i)
2: $\mathcal{L} \leftarrow \{L \mid L \text{ satisfies (8.9)–(8.16)}\}$
3: Find a distribution $\{p_L \mid L \in \mathcal{L}\}$ that conforms to the constraints (8.2)–(8.4) and maximizes the average sensitivity (8.5).
4: Sample L from $\mathcal{L}$ according to the probability distribution p_L
5: **return** L

interleaved list becomes over-constrained more often than for the conventional OI method: it happens about 10% of the time for VA-OI versus less than 0.1% for OI (experiments with the real rankings introduced in Section 8.5.1). In cases where the problem becomes over-constrained, we relax it by replacing constraint (8.4) by the aggregated version:

$$\sum_{k=1}^{N} \sum_{i=1}^{|\mathcal{L}|} p_i \delta_k(L_i) = 0. \tag{8.17}$$

By doing so we guarantee that the problem has a solution. Note that A and B are always present in $\mathcal{L}$, so we have at least two variables (p_1 and p_2) and at most two linear equations (8.3) and (8.17). We can also see that the interleaving method stays unbiased for the randomly clicking user, which we also confirm experimentally in Section 8.8.

We also considered altering the credit function to account for the fact that we are dealing with aggregated result pages. Vertical documents are often visually more salient than organic web documents, which changes the examination probabilities and in some cases even the examination order. We take this into account by redefining the $\mathrm{rank}(d, X)$ function in (8.8) as $\mathrm{rank}(d, \mathrm{examination}(X))$, which represents the rank of document d in ranking X, where X is reordered in descending order by examination probability according to a click model. Preliminary experiments with this adapted credit function showed that it makes no change for the sensitivity and correctness as measured with the offline and online reference metrics (Sections 8.7.2, 8.7.3). We did see a difference in sensitivity if Pareto dominance is used as ground truth (Section 8.7.1), but we treat this result as an artifact of our experimental setup, one that stems from the fact that both click model and Pareto dominance rely on the same examination order implied by the same click model.

To summarize, we have introduced two interleaving methods that respect vertical blocks within aggregated search scenarios. In Sections 8.6, 8.7, 8.8 we evaluate these methods experimentally and show how they compare to conventional interleaving methods.

8.5 Experimental Setup

Each of the interleaving methods we study (Algorithms 8.1, 8.2, 8.3, 8.4) takes two ranked lists as an input, generates an interleaved list, observes user clicks on it and infers which system produces better rankings. Therefore, in order to answer the three research questions listed in the introduction (Section 8.1), we need two ingredients: pairs of ranked lists (rankings) and clicks. Below we present different ways to obtain ranked lists and

Table 8.1: Characteristics of different experimental setups: documents, queries, verticals and aggregated search systems.

Data	Documents	Queries (TREC Topics)	Verticals	Aggregated Search Systems	Described in	
					rankings	clicks
real	two months of log data	814 (5,755*) queries	1 vertical	2 systems	8.5.1	8.5.4
model	TREC FedWeb13 data [60]	50 topics	12 verticals†	36 model systems	8.5.2	8.5.5
synthetic	artificial documents with simulated relevance	any number of synthetic topics	3 verticals	any number of synthetic AS systems	8.5.3	8.5.5

* We have more queries for the relaxed setup (see Section 8.5.4).
† Verticals are manually classified from the 108 FedWeb sources. Details of the classification can be found in [174].

clicks that we will later use in our experiments. The breakdown of the experimental setups that we use is presented in Table 8.1. We also present a table showing which experiments use which datasets (Table 8.2). Basically, we aim to choose the most appropriate dataset(s) that provide most valuable insights to answer each specific experiment we perform.

Rankings. We use three sources of rankings: from *real rankings* of a commercial search system through *model rankings* emitted by rather simplistic model *aggregated search* (AS) systems to completely *synthetic rankings* targeted to approximate realistic AS result pages while having more control.

The reason to use different setups lies in the need to find a trade-off between *reality*, *variability* and the *amount* of data we have (see Figure 8.3). While desirable, we are not in possession of a single setup that would cover all the aspects (dotted line in Figure 8.3). Instead, we have three different setups. Firstly, *real rankings* allow us to assess interleaving methods on real-world ranking systems that serve real users and give us access to a good amount of data. However, variability of the data is low (e.g., we only have access to a single vertical and hence cannot assess our interleaving methods in a multi-vertical environment). Secondly, *model rankings* represent near real-world ranking systems, which, however allow for greater variability, since we use simulated users and do not run the risk of hurting real users' experiences here. The data, however, is limited by the number of model AS systems (36 in our case) and the number of TREC FedWeb topics (50 in our case). And lastly, *synthetic rankings* allow us to generate endless amounts of data with

Table 8.2: Experiments performed to answer each research questions and the datasets used.

Research Question	Experiment	Section	Data		
			real	**model**	**synthetic**
RQ5.1 influence on the user	visual aspects	8.6.1	✓	✓	✓
	offline quality	8.6.2		✓	
	online quality	8.6.3	✓	✓	
RQ5.2 correctness and sensitivity	Pareto dominance	8.7.1		✓	✓
	offline metrics	8.7.2		✓	
	online metrics	8.7.3		✓	
	A/B-testing	8.7.4	✓		
RQ5.3 unbiasedness	random clicks	8.8.1	✓	✓	✓

any variability we want, but the relation to the real world is limited to what we explicitly include in our synthetic ranking generation procedure.

Clicks. For clicks, we use both *real click* data from a two-month click log (see below) and *simulated clicks* derived from a state-of-the-art click model for aggregated search. The click log setup is closely tied to the ranking setup: the real clicks can only be used with the real rankings, while the other ranking setups require modeling the user clicks with a click model.

8.5.1 Real Rankings

We use a two-months click log of the Yandex search engine recorded during winter 2012–2013. Due to limited access to the click log data we were only able to collect data with one vertical block. We picked the vertical of Mobile Applications since it is visually different from normal web results and at the same time does not have additional aspects such as freshness [65, 96] (so we ruled out the News vertical) or horizontal document layout within a vertical block (that ruled out the Image and Video verticals). The Mobile Applications vertical is triggered for queries with an intent related to smartphone or tablet applications.[8] While the algorithm used for intent detection is beyond the scope of this study, it is useful to name a few examples of queries triggering the appearance of this vertical. These include queries containing a mobile application name (e.g., `[cut the rope]`) or explicit markers of mobile platforms (e.g., `[Opera for iPhone]`). The result items in the vertical block are presented using a standard text snippet, enhanced with an application icon, price and rating. An example of such a snippet is shown in Figure 8.4.

One of the limitations of this approach is that we cannot experiment with completely new document orders that are disallowed by the current production ranking algorithms.

[8] Limited to iOS or Android devices at the time of collecting data.

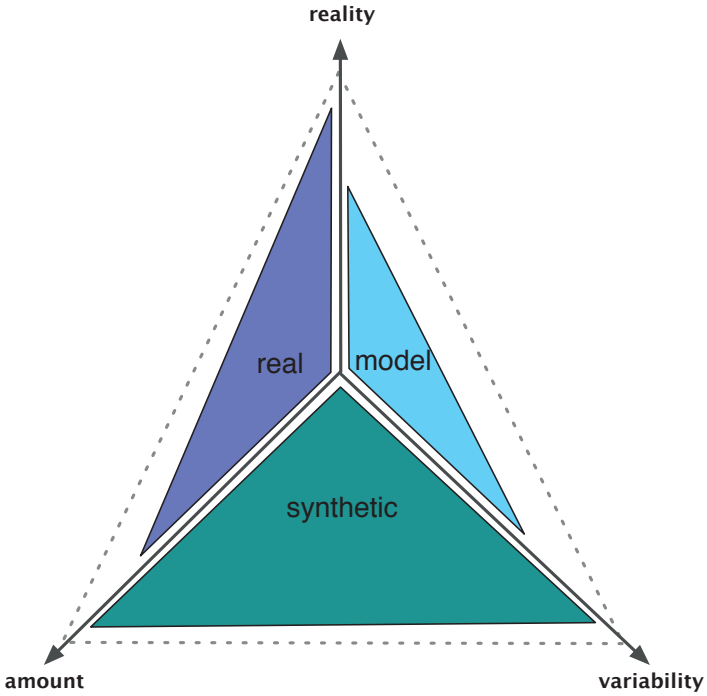


Figure 8.3: Radar chart of the different data characteristics covered by the different experimental setups (*real*, *model* and *synthetic* data). The further from the center, the more pronounced this property is in a particular data set. Note that for illustrative purposes, the intersection of the three axes denotes that all the characteristics are low, but not zero. E.g., *model* data exhibits relatively high variability and reality with amount of data being low.

In particular, we cannot reproduce the outcomes of an interleaving method that does not respect vertical block grouping. Another limitation is that we do not have data with multiple verticals and hence can only reproduce part of the analysis that we are able to conduct with model or synthetic rankings introduced below. On the other hand, even limited experiments with real data allow us to validate the key findings of the more comprehensive but less realistic setups.

8.5.2 Model Rankings

A broader but less realistic scope of rankings can be obtained by implementing model *aggregated search* (AS) systems. As in [174], we implement near-real-world aggregated search systems that we can apply to rank documents and emit rankings with vertical blocks. The basic idea is that we implement state-of-the-art aggregated search components and by combining different components, we develop a set of aggregated search pages with varying qualities.

Firstly, we assume that an aggregated search page consists of ten Web blocks (a single

Figure 8.4: A vertical result item from the Mobile Applications vertical for the query [learn languages for free iPhone].

organic web document is viewed as a block) and up to three vertical blocks dispersed throughout those ten blocks (where each vertical block consists of a fixed number of three items).

Secondly, as described in previous work [174], we simulate a set of systems by implementing existing algorithms. We develop four state-of-the-art *vertical selection* (VS) systems; two utilize vertical term distribution evidence and the other two use user intent information by mining query logs. For IS we simulate three potentially different levels of relevance by using different ranking weighting schemes: Perfect (all vertical items are relevant), BM25 (with PageRank as a prior) and TF (with PageRank as a prior). We also simulate three result presentation approaches for *result presentation* (RP): Perfect, Random and Bad. Perfect places the vertical blocks on the page so that gain could potentially be maximized, where Bad reverses the perfectly presented page.

By utilizing this approach, we can generate $4 \times 3 \times 3 = 36$ aggregated search systems, which gives us $36 \times 35/2 = 630$ system pairs.

As a source of documents we use the TREC FedWeb13 dataset [60], which has topical relevance labels as well as vertical orientation labels, i.e., the probability that the user prefers documents of this vertical type to the organic web documents.

8.5.3 Synthetic Rankings

While the model AS systems allow us to have broader variety and a larger amount of data than real rankings, they still provide a limited setup. In order to make reliable judgements about the sensitivity of an interleaving method (**RQ5.2**) or about its unbiasedness (**RQ5.3**) we would like to be able to generate as much data as we need. For this purpose we introduce a synthetic ranking generation procedure that does not require any document dataset or pool of AS systems.

First of all, we want to mention that two ranking systems, when compared, may or may not differ in how they present vertical documents. In particular, the systems being compared may use the same method of building and presenting the vertical blocks and only differ in the non-vertical (organic) ranking. We believe that this is an important special case for search engine evaluation and it deserves a separate analysis. Our early experiments with a real click log [40, Table 4] showed that the outcome of the interleaving method is correlated with, but different from, the interleaving in situations where no verticals are distorting user clicks.

For the synthetic experiments we design two algorithms to simulate rankings; those with *fixed* vertical blocks (position and contents of the vertical blocks is the same in A and B rankings) and *non-fixed* vertical blocks (vertical blocks may have different contents and positions in A and B). These two settings also correspond to two approaches to the

construction of aggregated search page (Section 2.2). The first approach [7, 8] assumes that we have an organic web ranking in place and that vertical documents are being added on top of it, regardless of what the web ranking is. Another approach [154, 174] assumes that the vertical documents also have topical relevance labels that can be compared to those of the web documents. This approach has started gaining popularity with the recently established TREC FedWeb track [60, 120].

For the *fixed* vertical blocks we first generate a pair of organic rankings and then randomly insert blocks of vertical documents at the same place in both rankings. These blocks vary in size from zero to eight documents and in the positions of the blocks.[9] The ranking of the vertical documents within a block is always fixed, i.e., the same for both rankings. Vertical documents are either (1) non-relevant (condition *non-relevant*); or (2) a number of them, proportional to the relevant organic web documents, is relevant (condition *relevant*).

For the *non-fixed* vertical blocks we generate two lists that contain vertical documents straight away. We slightly modify the above procedure that we use for organic ranking generation: when constructing the ranking of all documents, both vertical and organic web documents are generated; if the vertical documents end up not being grouped, we reorder them accordingly (see Algorithm 8.7).

When generating pairs of rankings A and B, we also ensure that the difference between a generated pair of rankings resembles the typical differences that we encounter in real-world interleaving experiments while allowing for ample variety. A particular procedure for generating synthetic ranking pairs is described in Appendix 8.10.

8.5.4 Real Clicks

One problem that we face with click data, is that we need to have clicks on the interleaved document list. If the particular interleaving method we want to study has not been deployed in production, we do not necessarily have the clicks that we need. To address this problem we adopt the setup proposed by Radlinski and Craswell [124] that makes use of historic user clicks in order to evaluate new interleaving methods. The main idea of the method is to look at queries with sufficient variation in the ranked lists and then pretend that these different ranked lists are the interleaved lists for some rankers A and B.

Let us call a query together with a result list a *configuration*. Each configuration that has been shown at least once to a user counts as an *impression*. Each impression has zero or more clicks. We proceed in the following steps:

1. Keep only impressions that have at least one click on their top-N documents.
2. Keep only configurations that have at least K such impressions.
3. Keep only queries that have at least M such configurations.

After that, we pick two configurations to be rankings A and B for each query. Following Radlinski and Craswell [124], we call the most frequent configuration *ranking* A and the one that differs at the highest rank *ranking* B. In cases when we have several candidates for B, we choose the most frequent one. Once we have our rankings A and B, we compute all possible interleaved lists that can be produced by Algorithm 8.3 and proceed with the filtering:

[9] We randomly assign block positions based on the distribution obtained from the TREC FedWeb data [60].

Table 8.3: Filtering parameters setup.

	N	K	M
Radlinski and Craswell [124]	4	10	4
our setup	10	4	2

4. Keep only queries for which we have all interleaved lists that can be produced by VA-TDI in the log.[10]

In order to fully define the experimental setup we have to define the parameters K, M and N. We summarize the parameters we use in Table 8.3. Unlike [124] we cannot use only the top-4 documents as this is highly likely to be in-between the vertical block. This is why we are forced to decrease K and M in order to have a sufficient amount of log data. With these parameters we have 814 unique queries. If we relax the last filtering step and only require at least one interleaved list to be present in our query log, we obtain $5{,}755$ queries to experiment with (we consider the missing interleaved lists as ties). The reason to consider this *relaxed* setup is the following: if we required all possible interleaved lists to be present in the click log, we would end up in a situation where only very similar rankings A and B are left, which is an additional bias we want to avoid.

The main limitation of this approach is that we have a very limited amount of data, which is not the case for the click simulation (Section 8.5.5). We should also take into account that the data we get using this method is skewed towards a relatively small number of highly frequent queries. The variety of rankings is also limited by those extracted from the click log (Section 8.5.1)—we cannot combine real clicks with model or synthetic rankings.

8.5.5 Simulated Clicks

We simulate users' click behavior on an interleaved list using click models. Simulated users are always presented with the top ten documents from the interleaved list. We have two types of user simulations.

Random User The *random click model* (cf. Section 3.1.1) assumes that users click on each document in the presented ranking with probability $\rho = 0.5$, such that—in expectation—half the documents are clicked. Relevance or presentation of documents is not taken into account. This model is used to answer **RQ5.3**.

Federated Click Model The *multi-vertical federated click model* (mFCM) (see Section 5.5.1) is used in order to answer **RQ5.1** and **RQ5.2**. This click model is designed to capture user behavior when result pages contain vertical documents of different vertical types. It assumes that user attention may be attracted to vertical documents as well as documents adjacent to a block.

[10] For historical reasons we report only on VA-TDI, since it was the only interleaving method considered during the period when we had access to the click log.

This model is an extension of the FCM model used in [40] for the case of multiple verticals. Unlike the simple model of position bias used in [82], we also take into account the fact that vertical blocks and adjacent documents draw additional attention of the user, more so if the vertical orientation is high.

8.6 Influence on the User Experience

When one wants to evaluate an interleaving method, its effect on the user experience needs to be evaluated first. We formulate this as one of our research questions (**RQ5.1**) and conduct three experiments.

Firstly, we look at the visual changes of the interleaved list in Section 8.6.1. Secondly, we evaluate the quality of the interleaved list as captured by offline metrics (Section 3.2) and compare it to the quality of the systems being interleaved (Section 8.6.2). And finally, in Section 8.6.3 we perform a similar quality comparison using absolute click metrics (Section 3.3) based on data produced by real search engine users as well as by simulated users.

8.6.1 Visual Changes in Result Page

Setup

A good online evaluation method should be opaque to the user. That is, the user should not notice any interface changes when interacting with an interleaved list as opposed to a base ranking system. We identified several particularly prominent features of aggregated search systems related to vertical blocks, namely the number of vertical blocks per vertical and the size of a vertical block, and track how they change when we use different interleaving methods. If an interleaving method shows smaller or bigger vertical blocks or, especially, if it splits a vertical block into two (thus, increasing the number of blocks) we consider this to affect the user, which we want to avoid.

As a source of rankings we use *real*, *model* and *synthetic* rankings (see Sections 8.5.1, 8.5.2 and 8.5.3).

Results

Synthetic Rankings. We start with *synthetic* rankings (Section 8.5.3), since they allow us to vary the target block size of the ranking lists being interleaved. Namely, we can set the block size exactly if we use *fixed* vertical placement (Algorithm 8.6, each vertical has a block of the specified size). For *non-fixed* placement the target block size is an expected number of vertical documents per block, specified using the distribution q in Algorithm 8.7 (i.e., for target block size 2 we set $q_t = 2/10$ for each vertical t). We also considered the case of up to one vertical type and a multi-vertical case where up to three verticals are allowed. In the latter case, the smaller target block sizes are considered, since a result page of ten documents cannot have more than ten vertical documents in Algorithm 8.7.

Figures 8.5 and 8.6 show our results for different numbers of blocks per vertical[11] and Figures 8.7 and 8.8 for vertical size. For VA-TDI, we see that the number of generated

[11] Figure 8.5 differs from the corresponding figure in [40] (Figure 4 there) due to different settings of the

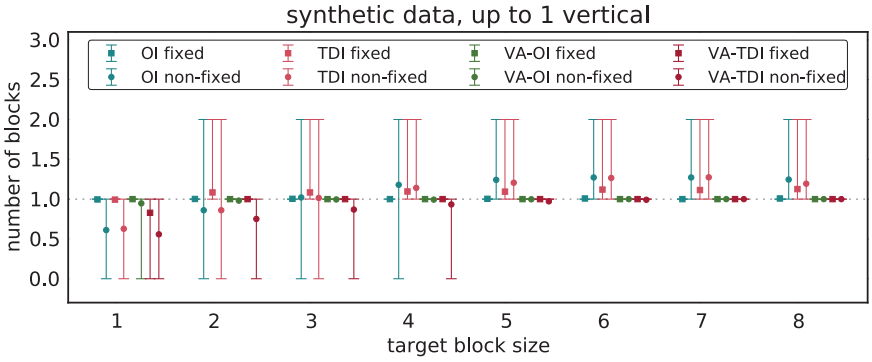


Figure 8.5: Average number of vertical blocks per vertical; target block sizes 1–8, up to one vertical type. Error bars correspond to 5th and 95th percentiles.

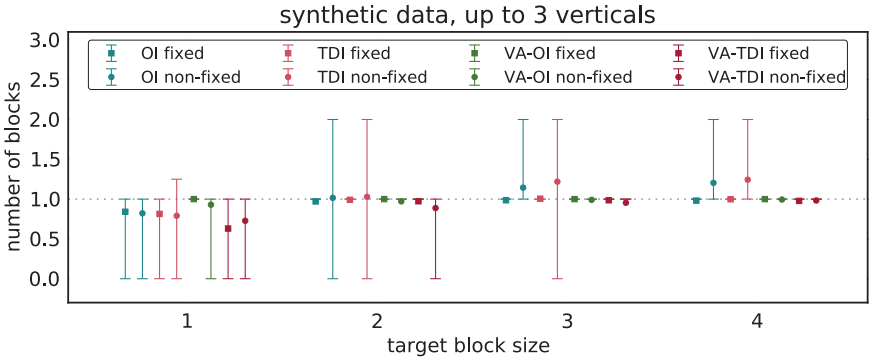


Figure 8.6: Average number of vertical blocks per vertical; target block sizes 1–4, up to three different vertical types. Error bars correspond to 5th and 95th percentiles.

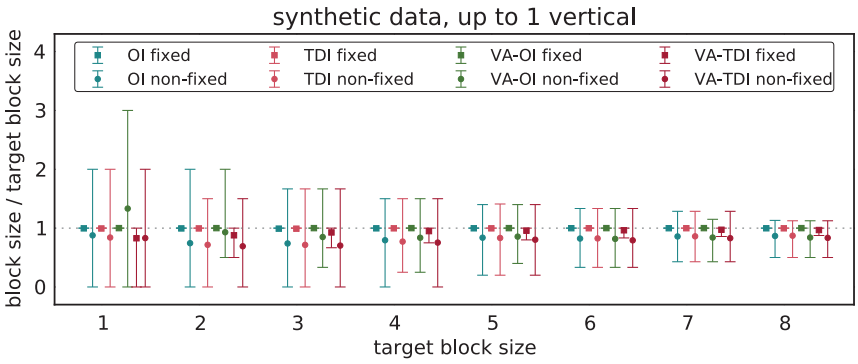


Figure 8.7: Average vertical block size divided by the target block size; target block sizes 1–8, up to one vertical type. Error bars correspond to 5th and 95th percentiles.

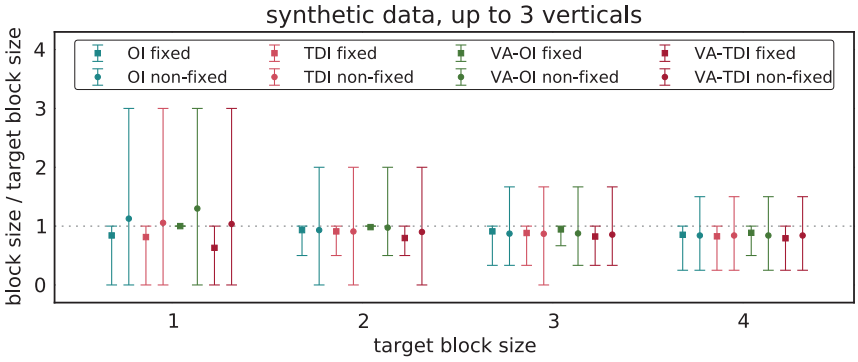


Figure 8.8: Average vertical block size divided by the target block size; target block sizes 1–4, up to three different vertical types. Error bars correspond to 5th and 95th percentiles.

vertical blocks is typically close to and never higher than 1, as designed. Due to equations (8.14) and (8.15), VA-OI always has exactly one vertical block per vertical if the *fixed* placement scheme is used. When the vertical blocks in the lists we interleave are small, the block may not be included in the interleaved list at all, resulting in an average number of blocks smaller than one (cf. Algorithm 8.3, line 22). This can also occur when the vertical blocks are placed in the lower halves of the original lists. For TDI, we observe different behaviors under *fixed* and *non-fixed* block placement. When the rankers we compare have different vertical blocks (condition *non-fixed*), TDI tends to generate several smaller blocks which result in a higher average number of blocks. The same goes for OI.

All interleaving methods behave similarly with respect to block size. When two verticals have the same vertical block (*fixed* block placement), the block size can only become smaller after the ranked lists they are contained have been interleaved (values less than 1 in Figures 8.7 and 8.8) because some vertical documents are pushed outside the top ten. This effect is more visible when multiple vertical blocks from multiple verticals are present (Figure 8.8). When the verticals are generated using the *non-fixed* scheme (Algorithm 8.7), the resulting block size to target block size ratio on average is the same as, or even smaller than, in the *fixed* scheme. It also has a wider distribution. The reason is that the target block size is not always equal to the exact size of the corresponding vertical block in A and B rankers; in fact, A and B can have different block sizes.

Real and Model Rankings. Now we repeat the same experiments for the *model* and *real* datasets (Sections 8.5.2, 8.5.3). In those cases we no longer have control over the size, position or contents of the vertical blocks in the A and B rankers, so we simply plot the overall picture that includes blocks of different sizes.

Figure 8.9 shows the results for the *real* data. We can see that VA-TDI only has slightly less than one block per vertical on average, while it never exceeds 1, i.e., the vertical block is not broken. We can also confirm that VA-TDI results in smaller vertical blocks than the other interleaving methods (TDI, OI, VA-OI). VA-OI always yields one vertical block as is guaranteed by equations (8.14) and (8.15), and the fact that we only have one vertical in the real dataset.

Figure 8.10 reports similar results for the *model* data. We see the same pattern that VA-TDI tends to have smaller blocks and misses a block completely (number of blocks zero) more often than TDI and OI, which, in turn, often have a vertical block split up into two or even three smaller blocks. Now VA-OI also has less than one block per vertical on average, which is normal because A and B can have different verticals and not all of them should be present in the interleaved list. The difference between average number of blocks of OI and TDI is not observed for the real data, but is observed for the model data which suggests that the difference between these methods depends on the ranking systems being compared and number of different verticals present in the dataset (remember that real data rankings have only one vertical). Also, vertical-aware methods VA-TDI and VA-OI are more conservative about the vertical blocks, and yield smaller blocks and fewer blocks than conventional interleaving methods. We believe this to be less visible and to have a

synthetic generation procedure. Previous results can be obtained if we set $d = 0$ and $\tau = 0$, i.e., two generated rankings are just two random permutations of the same documents. We believe, however, that the settings we use here (see Appendix 8.10) are more realistic.

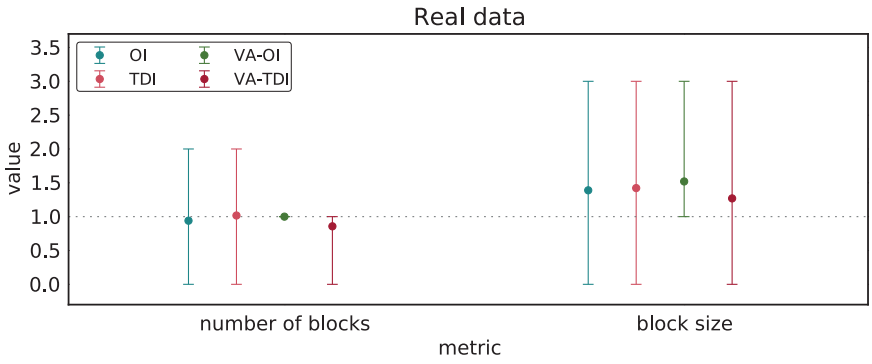


Figure 8.9: Average number of vertical blocks per vertical and the vertical block size of each vertical; real data. Error bars correspond to 5th and 95th percentiles.

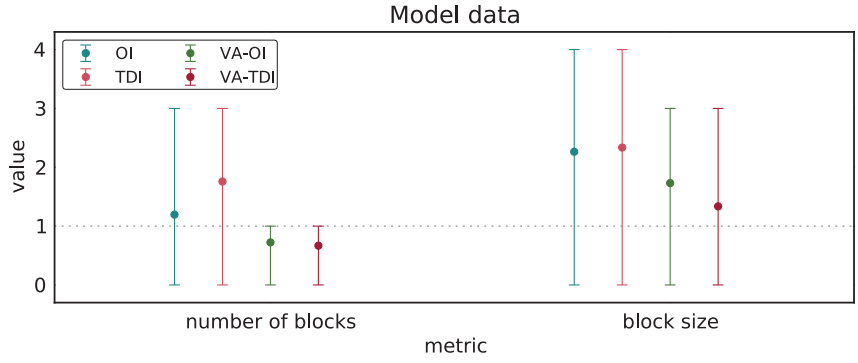


Figure 8.10: Average number of vertical blocks per vertical and the vertical block size of each vertical; model data. Error bars correspond to 5th and 95th percentiles.

less disturbing effect on the user experience than splitting the block of the same vertical type.

We conclude that VA-TDI and VA-OI keep vertical documents together, as designed. These methods produce up to one vertical block per result list, thus bounding the impact of interleaving on the user experience. When vertical blocks are placed independently, the impact of TDI and OI without vertical awareness is high. However, when blocks are placed at the same *fixed* positions, the impact is much lower, especially for OI (but still higher than for its vertical-aware extension VA-OI).

8.6.2 Offline Quality Measures of the Interleaved Page

Setup

Our second and third sets of experiments involve comparing an interleaved system's performance to that of the systems A and B that are being interleaved. Ideally, we want an interleaving method to produce ranked lists that are not worse than those of A and B. We see it as one of the main contributions of our work and claim that every new interleaving method should be tested for that.

In this section concerning offline quality measures we only use *model* rankings (Section 8.5.2) since these rankers, unlike *synthetic* rankers, have topical relevance and vertical orientation labels that we can use for quality comparison.[12] We do not use *real* rankings since only a small fraction of these documents have relevance judgements.

Given two rankings A and B, we can compute a relevance-based quality measure for A, B and for the interleaved lists similar to He et al. [79]. As our quality metrics we use a classic evaluation metric NDCG [89] (Section 3.2.5) as well as a metric specific to aggregated search, AS_{RBP} [171]. Separately, we also analyse simple one-component aggregated search metrics [174] to verify that our interleaving procedure does not lead to degradation in any of the aspects of aggregated search (VS, VD, IS, RP). The choice of AS_{RBP} is motivated by the fact that this metric is the best at capturing all four aggregated search aspects (provided that they are all treated equally important) and at discriminating different aggregated search systems [174]. The simple single-component aggregated search metrics we use are adopted from the aggregated search meta-evaluation study by Zhou et al. [174]; they independently reflect basic aggregated search properties and are as simple as possible in order to be agnostic to potential differences in the interleaving methods (e.g., the credit function choice). The same metrics were used in Section 5.3 (page 50).

For each offline metric M and each pair of 36 model rankers (Table 8.1) we first determine which ranker in the pair has a higher average value of M. This ranker is named A and the other one B. Ideally, our interleaved system should have an average value of quality metric M that lies in between those of A and B, respectively. We also perform a paired t-test (two-tailed 95%) to see how many interleaving experiments (out of $C_{36}^2 = 630$ experiments corresponding to pairs of model rankers) result in an interleaved systems being statistically significantly worse than B (i.e., the worst out of

[12] More details on the topical relevance and vertical orientation labels available for the *model* rankings can be found in [171].

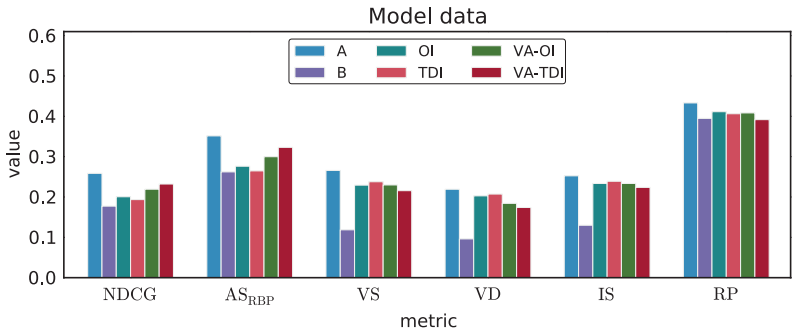


Figure 8.11: Average value of offline quality measures for the interleaved lists and original rankings A (better system) and B (worse system).

Table 8.4: Percentage of pairwise comparisons (out of 630) where the interleaved ranking scores statistically significantly lower than B as measured by the offline quality measures.

	NDCG	AS_{RBP}	VS	VD	IS	RP
OI	9.5%	13.2%	0.0%	0.8%	0.0%	0.5%
TDI	11.6%	21.4%	0.0%	3.3%	0.0%	0.2%
VA-OI	0.2%	0.8%	0.5%	0.3%	0.2%	0.5%
VA-TDI	0.2%	0.2%	0.0%	0.6%	0.2%	3.0%

two systems). We do not perform multiple testing corrections since we are not interested in the absolute values here. Instead, we compare the number of detected significant differences for different interleaving algorithms: if the null hypothesis is indeed true (i.e., two systems have the same quality) the odds of erroneously reporting N significant differences decreases with N (exponentially if we assume that the tests are independent).

Results

Figure 8.11 and Table 8.4 summarize our results. All interleaving methods, on average, perform better than B and worse than A. It is also worth noting that VA-TDI and VA-OI perform substantially better than TDI and OI if measured by the classic NDCG metric. We believe this to be an artifact of the model dataset (Section 8.5.2) that we are using. We noticed that in general, there are more relevant organic web documents than there are relevant vertical documents [60] and, since vertical-aware methods produce smaller vertical blocks (Figure 8.10) and those blocks tend to be lower in the ranking than they are for TDI and OI, we conclude that this leads to higher NDCG scores.

Table 8.4, in particular, tells us that only a small fraction of the model systems, once interleaved using VA-TDI, would suffer from quality degradation. The same holds for VA-OI, although degradation in terms of AS_{RBP} is slightly higher. On the other hand, we will much more likely have a quality degradation for a big portion of system pairs if we use OI and even more likely if we use TDI. As to the one-component metrics, VS, VD, IS

and RP, we only notice a small degradation of vertical diversity (VD) of the TDI rankings (possibly due to the fact that some verticals are pushed out of the top ten documents presented to the user) and result presentation (RP) of the VA-TDI rankings.

In summary, when evaluating offline, we conclude that our VA-TDI and VA-OI methods outperform TDI and OI in preserving the user experience.

8.6.3 Online Quality Measures of the Interleaved Page

Setup

Next we repeat the experiments from the previous section, but instead of computing offline relevance-based metrics, we compute online click-based metrics. This type of analysis is less precise, but can be used in situations when we do not possess relevance labels. This time we use *real* and *model* datasets. For the *real* dataset (Section 8.5.1) we compute click metrics for all impressions of all interleaved lists L that appear in our click log (Section 8.5.4). For the *model* dataset (Section 8.5.2) we simulate user clicks using the mFCM click model (Section 8.5.5), repeated 50 times. The *real* dataset allows us to work with real user clicks, whereas the *model* dataset gives us variability. Since the amount of data is not an issue here, we do not need to use *synthetic* data in this experiment (cf. Figure 8.3).

As absolute metrics we use the metrics that are often used in A/B-testing experiments. We decided to use metrics that only require clicks and no additional information (like relevance judgements, user information, timestamps or session information): Clicks@1, MaxRR, MeanRR, MinRR, PLC, see Section 3.3. These metrics were also used in the interleaving study by Chapelle et al. [27]. We also add one vertical-specific metric *VertClick*, which equals 1 if there was a click on a vertical document and 0 otherwise. Unlike previous metrics, this one measures less of the quality of the result page, but rather how much attention is being drawn to the vertical documents.

Results

Real rankings. The results for the *real* dataset are summarized in Figure 8.12. As we are not interested in the absolute values of the metrics (and are not able to disclose them due to the proprietary nature of such information), we normalize all the metrics to the corresponding values of the system A which also happens to perform better than B according to all those metrics.

We can see from the plot that the click metric values for the interleaved list L are between those of A and B. The differences between them are not statistically significant when using a 95% paired two-tailed t-test. We conclude that we do not have any degradation in user experience compared to the worst of two systems (B in this case). We should mention that we do have a degradation (not significantly) compared to the best system (A), but this cannot be avoided since we do not know which system is superior beforehand since finding this is the purpose of performing the evaluation in the first place.

Model rankings. Figure 8.13 shows that, on average, every interleaving method produces a system that scores between A and B according to all online metrics. We can

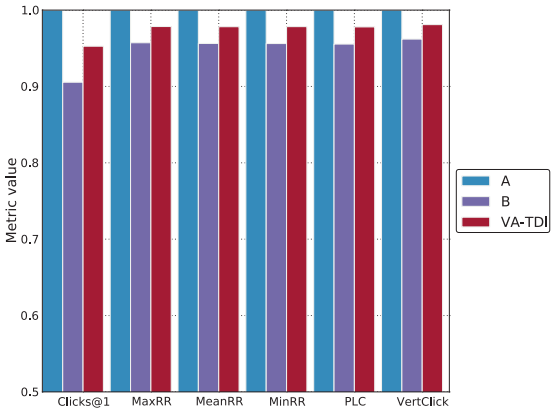


Figure 8.12: Normalized scores for online click metrics for the rankings A and B and for the interleaved list obtained using VA-TDI (*real* dataset).

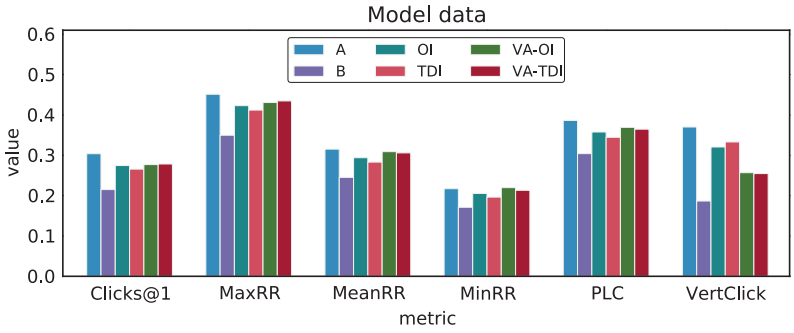


Figure 8.13: Average online click metric scores for the rankings A and B and for the interleaved list obtained using different interleaving methods (*model* rankings, *simulated* clicks).

Table 8.5: Percentage of pairwise comparisons (out of 630 model ranking pairs) where the interleaved ranking scores statistically significantly lower than B as measured by online quality measures.

	Clicks@1	MaxRR	MeanRR	MinRR	PLC	*VertClick*
OI	7.1%	1.7%	4.9%	5.7%	4.3%	4.0%
TDI	7.6%	3.2%	6.8%	10.0%	8.1%	4.0%
VA-OI	4.9%	1.1%	0.8%	0.3%	0.5%	20.0%
VA-TDI	4.6%	0.5%	0.8%	1.6%	0.8%	27.9%

see that VA-TDI and VA-OI are slightly better than TDI and OI according to all metrics except *VertClick*, where VA-TDI and VA-OI have substantially lower scores. This is also supported by Table 8.5; like Table 8.4 it shows how many comparisons result in statistically significantly lower metric values for the interleaved system using a paired two-tailed 95% t-test. One explanation for the low *VertClick* values for VA-TDI and VA-OI is that they tend to produce smaller blocks (see Figure 8.10), so that we have fewer vertical documents and therefore they have a lower probability of being clicked.

In this section we presented a thorough analysis of the influence that an interleaving method has on the user experience, on both visual and quality aspects. Such an analysis should be at the core of evaluating any new interleaving method, even though it was largely ignored until now. In answer to **RQ5.1** about the influence of interleaving on the user experience, we demonstrated that the newly-introduced vertical-aware interleaving methods not only preserve the user experience, but they also preserve the quality of the ranking systems being interleaved, with the exception of the controvertial *VertClick* metric.

8.7 Correctness and Sensitivity

Our second research question **RQ5.2** concerns the amount of data we need to be able to draw conclusions about the system quality (*sensitivity*) and how accurate this conclusion is (*correctness*). Typically, before a new ranking algorithm is launched to the public, it is compared to the previous version of the algorithm using interleaving or some other method. It is therefore crucial that we can quickly and accurately decide if the new ranking algorithm is better or worse than the current one.

In Sections 8.7.1, 8.7.2 and 8.7.3 we analyze how the fraction of *correctly* identified preference pairs depends on the number of user impressions. We first look at the strong cases where one ranker *Pareto dominates* the other (Section 8.7.1) and then analyze more relaxed settings where instead of Pareto dominance we use offline quality metrics like NDCG or AS_{RBP} (Section 8.7.2) or online metrics like MeanRR or Clicks@1 (Section 8.7.3). In Section 8.7.4 we consider typical ten-day experiments and see how often interleaving methods can come to a statistically significant decision in this time frame and how they compare to A/B-testing in terms of *sensitivity*.

8.7.1 Finding Strong Differences: Pareto Dominance

Setup

Our simulation approach is based on the experimental setup proposed by Hofmann et al. [80, 83]. We first take two *synthetic* (Section 8.5.3) or *model* (Section 8.5.2) ranked lists, apply an interleaving method, and subsequently offer the interleaved list to a *simulated* user that produces clicks (Section 8.5.5). Then it is up to the interleaving method to select a winning ranker. Our experiment measures to what degree an interleaving method can detect differences in the quality of result lists. Specifically, we look at how the confidence about the preferred ranker depends on the number of user impressions (sensitivity) and what the final correctness level is. The fewer impressions are needed for an interleaving method to determine the winner, the more sensitive the method is.

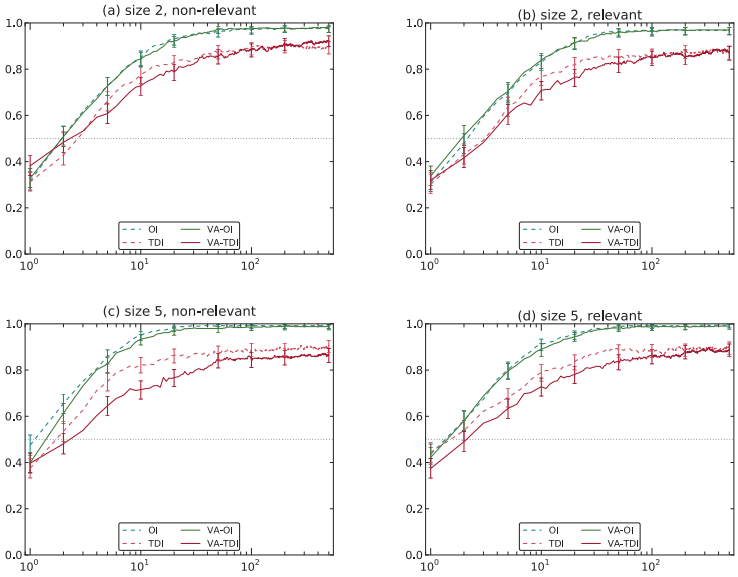


Figure 8.14: Portion of correctly identified ranker preferences (vertical axis) by different interleaving methods after 1–500 user impressions (horizontal axis, log-scale). The dashed horizontal line at 0.5 denotes random preference. All figures have independent block placement. Error bars correspond to 95% binomial confidence intervals. All rankings have one or zero verticals.

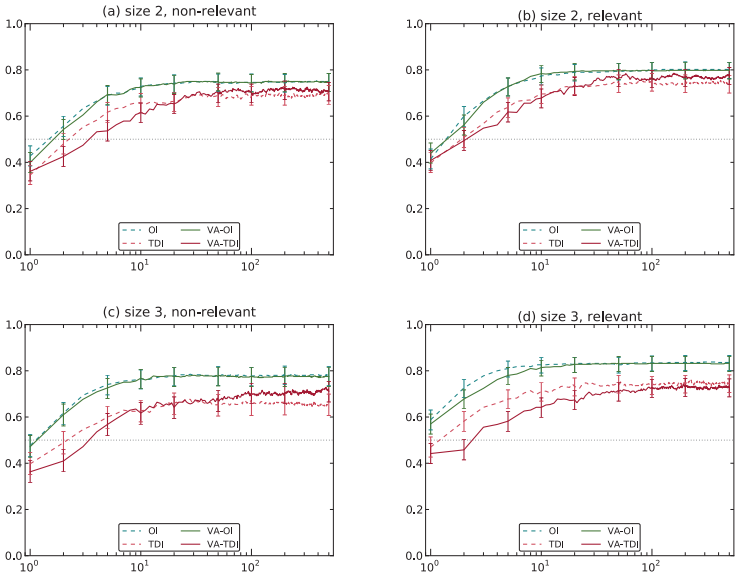


Figure 8.15: Same as above, but rankings can have up to three different verticals.

We repeat the process of generating two synthetic rankings (or randomly drawing a pair of model rankers and a query) until one ranking Pareto dominates[13] the other in terms of how it ranks relevant documents. Because the rankings are chosen in such a way that one dominates the other, we know which ranking should be preferred by an interleaving method.

We generate 500 pairs of rankings, with one ranking dominating the other, as described above. These pairs are each interleaved 500 times by different interleaving methods. For synthetic rankings we repeat this process for several combinations of the conditions described in Section 8.5.3. We observe the portion of correctly identified ranking preferences (i.e., the accuracy) by each interleaving method. We calculate the mean and 95% binomial confidence bounds.

Results

Synthetic Rankings We compare our vertical-aware interleaving methods, VA-TDI and VA-OI, to the non-vertical-aware baselines, TDI and OI, in terms of the accuracy of the identified ranker preferences. Figures 8.14, 8.15 show the portion of correctly identified ranker preferences by TDI, OI, VA-TDI, and VA-OI after 1 to 500 user impressions modeled by the federated click model mFCM (see Section 8.5.5).[14]

In Figure 8.14 we only consider cases where rankings are allowed to have verticals of one type (e.g., News) whereas in Figure 8.15 up to three different types of vertical are allowed.

Figure 8.14(a) shows results averaged over all possible positions of a block of size 2 (*non-fixed* block placement) under the assumption that none of the vertical documents are relevant. We see that TDI and VA-TDI converge to correctly identify about 90% of the true preferences, whereas OI and VA-OI converge to about 98%. There is no significant difference in the number of impressions between conventional and vertical-aware methods.

Figure 8.14(b) shows results in the setting with relevant vertical documents. Results are almost identical to the case of non-relevant vertical documents with only subtle differences. The difference is more visible when allowing more verticals (Figures 8.15(a) vs. (b)) — the final converged accuracy level is higher when there are relevant vertical documents.

VA-TDI initially requires more sample data than TDI: TDI is significantly more accurate when we have a very small number of impressions (less than ten). We believe that the reason for this is the noise added by vertical documents dropping out, the problem discussed at the end of Section 8.4.1. Since this is noise—and not bias towards either ranking—this levels out as the number of observed impressions increases. However, this need for more samples by VA-TDI is a small loss in efficiency for a method that preserves the original user experience as much as possible compared to TDI.

Figures 8.14(c) and (d) show results for the same conditions as (a) and (b), respectively, but with a block size of five instead of two. The only difference with Figures 8.14(a) and (b)

[13] As in [83], we re-rank documents by examination probability $P(E_r = 1)$. In [83], $P(E_r = 1)$ is implicitly defined by the *cascade click model*. In our case, it is dictated by the mFCM model (Section 8.5.5); we marginalize over attention bias, using (5.9) and (5.14). We say that ranking A *dominates* B if, and only if, re-ranked with $P(E_r = 1)$, A ranks all relevant documents at least as high as B and at least one relevant document higher than B.

[14] In all our experiments all the methods plateaued after several hundred user impressions.

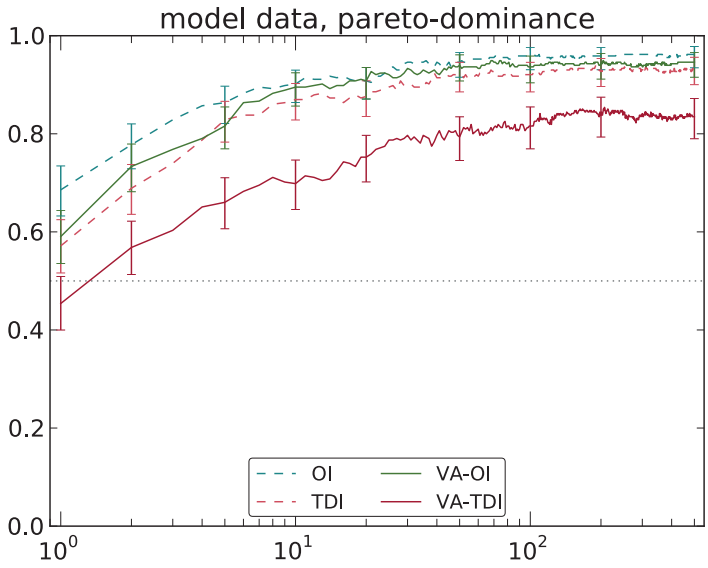


Figure 8.16: Portion of correctly identified ranker preferences (vertical axis) by different interleaving methods after 1–500 user impressions (horizontal axis, log-scale). The dashed horizontal line at 0.5 denotes random preference. Error bars correspond to binomial confidence intervals.

is the increased gap between TDI and OI families, which suggests that the latter should be preferred, especially in situations when we have many vertical documents. The same holds for Figure 8.15 where up to three different verticals are allowed.

Model Rankings Figure 8.16 shows the same experiment for the *model* data. This is a more realistic dataset than the synthetic one studied above. It contains vertical blocks of different types (see Table 8.1). From the figure we see that VA-TDI performs significantly worse than the other interleaving methods. As we explained at the end of Section 8.4.1, this is due to the fact that vertical documents often drop out of the interleaved list, thereby limiting the sensitivity of VA-TDI. This is especially true if the difference between lists being interleaved is large, which is often the case for the model data.

8.7.2 Finding Weak Differences: Offline Metrics

Setup

In this section we use a setup similar to the previous section, but instead of requiring that one ranker Pareto dominates another, we only require a non-zero difference in terms of an offline quality metric. This resembles the so-called "live data" simulations by Hofmann et al. [82] where interleaving methods are compared in terms of how well they can predict

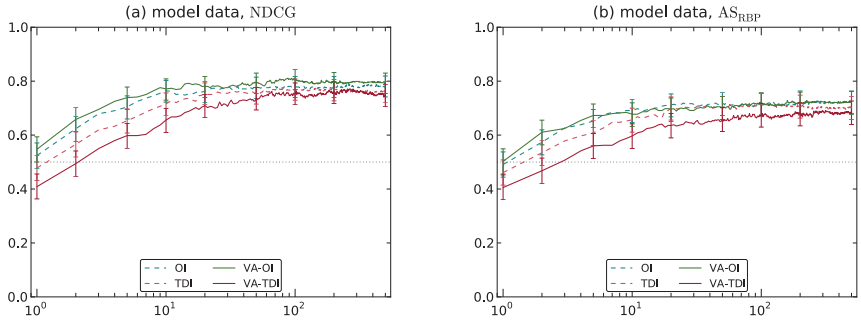


Figure 8.17: Portion of ranker preferences that agree with the offline metric (vertical axis) by different interleaving methods after 1–500 user impressions (horizontal axis, log-scale). The dashed horizontal line at 0.5 denotes random preference. Error bars correspond to binomial confidence intervals.

the direction of NDCG preference. On the one hand, offline metrics are less reliable as reference metrics than Pareto dominance. On the other hand, by using offline metrics we evaluate our interleaving methods' ability to find weak differences between rankings as opposed to strong cases of Pareto dominance.

We picked NDCG and AS_{RBP} to use in our experiments. We also experimented with the one-component aggregated search metrics (VS, VD, IS, RP), but found that only IS differences can be identified by the interleaving methods; only for this one-component reference metric the agreement level is more than 0.5. This is due to the fact that the one-component metrics are too simple to be used as reference metrics for system preference. One system can be better according to a one-component metric, but worse overall. Interleaving, in turn, considers the ranked list as a whole and is not required to agree with one-component metrics.

Here we only use *model* rankings (Section 8.5.2) since this is the only dataset that has meaningful relevance and vertical orientation labels.

Results

Results are presented in Figure 8.17. The error bars are big and it is hard to state that some method is more accurate than another, with the exception of VA-TDI, which converges much slower than VA-OI. In fact, for the first ten impressions with the NDCG reference metric (Figure 8.17(a)), the error bars do not overlap, indicating that VA-OI identifies significantly more correct ranker preferences than VA-TDI does. A similar picture is observed for AS_{RBP} (Figure 8.17(b)).

We also observe that all the interleaving methods agree better with the NDCG reference metric than with the AS_{RBP}. This demonstrates that it is more challenging to use interleaving methods to derive the preference of aggregated search pages than it is for homogenous pages. The loss of agreeing with AS_{RBP} might be due to the fact that the assumptions made in assigning credits of documents in interleaving methods align better with the NDCG position-based discount than with the more complex assumptions

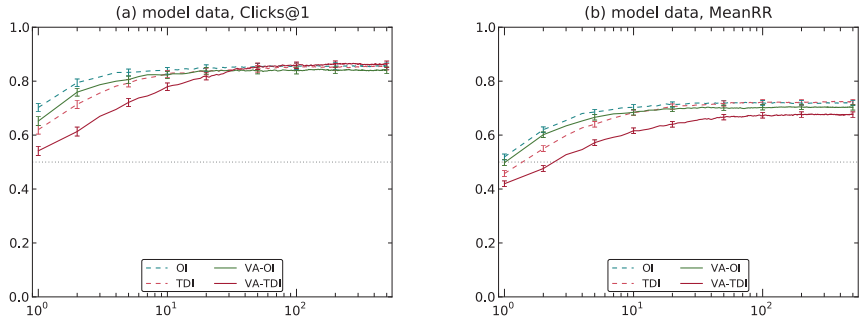


Figure 8.18: Portion of ranker preferences that agree with the online metric (vertical axis) by different interleaving methods after 1–500 user impressions (horizontal axis, log-scale). The dashed horizontal line at 0.5 denotes random preference. Error bars correspond to binomial confidence intervals.

of AS_{RBP}. It may also stem from the fact that the mFCM click model we utilize still assigns more value to topical relevance than to vertical orientation: equations (5.10)–(5.11) guarantee that only topically relevant documents are clicked whereas vertical orientation simply increases the chances of examining some documents.

8.7.3 Finding Weak Differences: Online Metrics

Setup

In this section we re-use the setup from the previous section, but use online click metrics as reference metrics instead of offline metrics: we identify the better ranking by comparing average values of an online metric from 50 query sessions, simulated using the mFCM click model (Section 8.5.5). We picked two online metrics: the very simple Clicks@1 metric, and position-aware MeanRR. These are also the metrics that, as we will see later, do not have statistically significant disagreement with interleaving in the real A/B-testing settings (Section 8.7.4). Since the metrics values now vary more, we increased the number of comparisons to 10,000 to reduce the churn due to randomness present in the metrics. This also helped us to overcome problems stemming from the fact that Clicks@1 is a binary metric and hence is often the same for A and B (we exclude such cases).

As in the previous section we only use *model* rankings (Section 8.5.2), because the click models heavily rely on the relevance labels and it is crucial for these relevance labels to be meaningful. Our preliminary experiments with *synthetic* rankings showed that the click metrics are very noisy in that case and all the interleaving methods only marginally agree with them.

Results

Results are shown in Figure 8.18. We see that all interleaving methods converge to the same level of accuracy of about 0.85 as measured by the Clicks@1 reference metric.

The agreement between the more complex MeanRR reference metric and the interleaving methods is less, but it confirms the previous finding (Figure 8.16) that VA-TDI is significantly less discriminative than the other interleaving methods.

8.7.4 Time-constrained Experiments and A/B-testing

Setup

In a real-world setting of a search engine evaluation experiment, the experiment is run for a certain period of time within which our interleaving method needs to reach a conclusion. In this section we reproduce such a time-constrained experiment setting using our *real* dataset (Sections 8.5.1, 8.5.4).

One issue of live experiments is that we do not have ground truth labeling of the better system. Usually, online click metrics are used to find the better system (e.g., in A/B-testing experiments), but these metrics may contradict each other or lack statistical significance. Therefore, one should not blindly use them as ground truth, but rather examine the full picture of their agreement and disagreement with the interleaving outcomes and with each other.

Given the click log data that we have access to (Section 8.5.4), we compute the following values for each query:

1. The average difference of the absolute click metrics for rankings A and B.
2. The interleaving score[15] for each impression of each ranking implied by VA-TDI. As some configurations might have more impressions in our click log than the interleaving method suggests, we normalize the scores to the correct probabilities as implied by the interleaving method. Specifically, we compute the average score for each configuration, multiply it by the probability of such a configuration and then average across all found configurations, similar to [124].
3. The interleaving scores for vertical documents and non-vertical (organic) documents separately, i.e., interleaving scores computed while only taking vertical (non-vertical) clicks into account.

In order to compare the direction of preference indicated by the absolute click metrics and the interleaving methods we split the data into six buckets corresponding to six equal time periods of ten days (t_1, t_2, t_3, t_4, t_5, t_6) and compute the weighted average of the absolute metrics and interleaving methods. The outcome for each impression (positive if A wins, negative if B wins, zero if it is a tie) is multiplied by the total frequency of the query[16] and summed up over all queries. We report the winning system according to each measure in Table 8.6. Note that we consider three ways of interpreting clicks in the interleaving method: *total*—all clicks are counted, *organic only*—only the clicks on non-vertical documents are taken into account, and *vertical only*—only the clicks on vertical documents are counted. For example, if we want to evaluate only changes in the organic ranking we may want to look at *organic only* results. On the other hand, we risk having an unbalanced team assignment if we skip the vertical block.

[15] We assume that the score is 1 if ranking A wins a particular impression, -1 if B wins and 0 if they tie.
[16] Remember that we previously normalized configurations to the correct probabilities.

Table 8.6: Agreement between A/B-testing measures and VA-TDI. All changes are statistically significant except for ones marked by ◇.

Measure	t_1	t_2	t_3	t_4	t_5	t_6
Absolute Metrics						
Clicks@1	B◇	B	B	A	A	B
MaxRR	A	B	B	A	A	B
MeanRR	A◇	B	B◇	A	A	B
MinRR	A◇	B	A◇	A	A	B
PLC	A	B	B◇	A	A	B
VertClick	B	B	B	B	B	B
VA-TDI (different variants)						
total	B	B	B	A	A	B
organic only	B	B	B	A	A	B
vert only	A	A◇	A◇	B	B	B

We also computed a per-query correlation between interleaving and A/B-testing metrics and performed a detailed analysis of the influence of the vertical block on this correlation (*total* vs. *organic only*). This analysis can be found in our previous work [40, Section 4.2.2].

Results

Table 8.6 shows that in most cases VA-TDI (*total* and *organic only*) agrees with the majority of the absolute metrics. Similar to what was reported in [127], the cases of disagreement between absolute click metrics and interleaving outcomes are always accompanied by the lack of statistical significance. We mark such cases where the winning system is *not* statistically significantly better with the ◇ sign.[17] We can also note that the agreement between the *vertical-only* VA-TDI and the *VertClick* absolute metric is low. However, in two of the three cases with disagreement VA-TDI does not detect a statistically significant preference. That means that either *VertClick* is too simple to correctly capture the ranking changes or *vertical-only* is not the right way to interpret interleaving outcomes (or both).

In this section we analyzed the sensitivity and correctness of conventional and vertical-aware interleaving methods under different settings and with different datasets. We showed that vertical-aware interleaving methods are in most cases as accurate as their conventional counterparts with the exception of VA-TDI which in some rare cases may be less sensitive than TDI because of vertical documents dropping-out. Some of our findings support the fact that OI and VA-OI are more sensitive and more accurate than TDI and VA-TDI, while the others suggest that they are at least as sensitive as TDI and VA-TDI. We also confirmed that interleaving methods are more sensitive than A/B-testing and demonstrated that these two approaches often disagree with each other.

[17] Here we use a bootstrap test with 1,000 bootstrap samples and a significance level of 99%.

8.8 Unbiasedness

Our third research question **RQ5.3** concerns unbiasedness of the interleaving methods. We investigate whether any of the interleaving methods identifies preferences for a ranker when there is no evidence in the data. For this purpose we employ randomly clicking users to show that there is no systematic bias in the algorithm.[18]

8.8.1 Randomly Clicking User

Setup

Our final simulated experiment assesses the unbiasedness of interleaving methods under random clicks (Section 8.5.5). It is important that accounting for vertical documents does not introduce bias, as it may otherwise lead to wrong interpretations of interleaving results. VA-TDI and VA-OI (Algorithms 8.3 and 8.4 respectively) were designed to be unbiased under many forms of noise; here we validate that our implementation does indeed fulfill this requirement.

Under the *random click model* (Section 8.5.5) an unbiased interleaved comparison method should not systematically prefer either ranker, i.e., the rankers should tie in expectation. We measure this following the methodology proposed by Hofmann et al. [83], by counting the number of comparisons for which a method detects a significant preference towards one of the rankers. For an unbiased method, this number should be close to the number expected due to noise. For example, a significance test with a p-value of 0.05 should detect statistically significant differences between rankers under random clicks in 5% of the comparisons. We repeat the test for different document datasets (*synthetic*, *model*, *real*) and for different numbers of impressions (from 100 to 500).

Results

Tables 8.7, 8.8, and 8.9 show the results, the percentage of detected significant differences, for synthetic (first two), model and real datasets, respectively. For all the methods and all datasets we see that the number of significant differences detected is in line with the expected 5%.

Using a one-tailed binomial confidence test (also with $p = 0.05$), we confirm that this number is only once significantly higher than 5%, in the case of OI with synthetic rankings and up to three verticals (non-relevant, fixed), see Table 8.9. We also see that the number of detected significant differences does not increase with the number of impressions which confirms that all the interleaving methods are, indeed, unbiased and can be relied on.

In this section we verified that none of the interleaving methods exhibit a bias under the random click model. We performed an analysis with different datasets and confirmed that there is no bias under an assumption of randomly clicking user. There could be other type of biases that we did not analyse in the current work.

[18] One may also test unbiasedness under other user models that are blind to results.

Table 8.7: Percentage of significant differences between rankers detected under the random click model for *model* rankings. $p < 0.05$ on 500 ranker pairs after 100–500 user impressions (left-most column). With $p < 0.05$, an interleaving method is expected to detect around 5% significant differences. None of the outcomes are statistically significantly higher than 5% using a one-tailed binomial confidence test (with significance level 0.05).

	OI	TDI	VA-OI	VA-TDI
100	4.4%	6.2%	6.2%	5.8%
200	3.8%	6.4%	6.4%	5.2%
300	6.0%	6.2%	5.2%	5.4%
400	4.8%	4.6%	4.8%	6.6%
500	4.4%	5.2%	4.8%	4.4%

Table 8.8: Percentage of significant differences between rankers detected under the random click model for *real* rankings. $p < 0.05$ on 500 ranker pairs after 100–500 user impressions (left column). With $p < 0.05$, an interleaving method is expected to detect around 5% significant differences. None of the outcomes are statistically significantly higher than 5% using a one-tailed binomial confidence test (with significance level 0.05).

	OI	TDI	VA-OI	VA-TDI
100	4.4%	4.0%	5.2%	6.0%
200	5.6%	7.8%	5.8%	4.2%
300	5.6%	5.8%	7.2%	5.4%
400	5.8%	6.2%	5.6%	5.4%
500	5.0%	5.0%	5.8%	5.0%

Table 8.9: Percentage of significant differences between rankers detected under the random click model for synthetic rankings. $p < 0.05$ on 500 ranker pairs after 100–500 user impressions (left-most column) under all combinations of conditions for block size 2. With $p < 0.05$, an interleaving method is expected to detect around 5% significant differences. (*) marks outcomes that are statistically significantly higher than 5% using a one-tailed binomial confidence test (with significance level set to 0.05).

up to one vertical

	non relevant								relevant							
	fixed				non-fixed				fixed				non-fixed			
	OI	TDI	VA-OI	VA-TDI	OI	TDI	VA-OI	VA-TDI	OI	TDI	VA-OI	VA-TDI	OI	TDI	VA-OI	VA-TDI
100	5.2%	4.0%	4.6%	4.6%	4.8%	6.2%	4.4%	5.0%	5.8%	5.4%	4.6%	5.8%	5.2%	6.0%	4.6%	4.0%
200	6.6%	5.6%	3.6%	6.0%	4.2%	5.4%	4.6%	6.6%	4.6%	3.0%	3.6%	2.6%	5.0%	4.6%	5.8%	6.2%
300	5.8%	5.8%	4.6%	4.4%	4.8%	5.2%	4.4%	4.4%	5.0%	5.0%	3.6%	4.0%	3.8%	5.4%	4.2%	5.6%
400	6.6%	5.6%	3.6%	3.6%	5.2%	5.0%	3.8%	5.4%	3.6%	3.4%	3.8%	5.0%	4.8%	5.2%	4.6%	7.4%
500	5.6%	5.2%	4.6%	4.4%	4.6%	5.2%	4.0%	5.0%	4.6%	5.0%	4.2%	5.0%	5.4%	4.4%	4.8%	5.6%

up to three verticals

	non relevant								relevant							
	fixed				non-fixed				fixed				non-fixed			
	OI	TDI	VA-OI	VA-TDI	OI	TDI	VA-OI	VA-TDI	OI	TDI	VA-OI	VA-TDI	OI	TDI	VA-OI	VA-TDI
100	5.2%	5.0%	4.6%	4.8%	3.4%	5.4%	5.0%	6.2%	4.6%	5.6%	5.8%	5.4%	5.6%	5.6%	3.6%	5.6%
200	4.6%	4.2%	4.2%	5.6%	3.6%	4.8%	5.0%	4.4%	4.2%	5.4%	7.0%	5.4%	6.2%	3.8%	4.6%	5.8%
300	6.4%	4.0%	4.4%	4.6%	4.2%	5.0%	5.2%	4.8%	4.4%	4.0%	5.0%	5.4%	6.2%	4.6%	5.2%	5.6%
400	6.2%	4.2%	5.6%	4.8%	4.4%	5.6%	5.0%	6.0%	4.8%	4.4%	5.0%	4.2%	5.0%	5.0%	5.4%	4.8%
500	7.4%*	4.4%	6.8%	5.4%	4.0%	4.2%	4.8%	4.6%	5.4%	4.4%	3.8%	3.6%	6.2%	4.0%	5.4%	4.6%

8.9 Conclusion

To assess different interleaving methods under a wide range of conditions, we have addressed three main research questions about the influence of interleaving methods on the user experience, about the correctness and sensitivity of interleaving methods, and about their unbiasedness, all with a special focus on vertical search. We have done so using both simulations and real rankings and clicks. With simulated rankings we have tested several vertical block sizes, several block placements and different levels of relevance within the block.[19]

To summarize, we have shown the following.

- Vertical-aware interleaving methods do not alter the user experience, as they do not break vertical blocks (Section 8.6.1) and do not degrade the quality of the result page (Sections 8.6.2, 8.6.3).
- Vertical-aware interleaving methods can find differences between rankings as fast and as accurate as their conventional counterparts with the optimized interleaving family being substantially more sensitive (Sections 8.7.1, 8.7.2, 8.7.3). We also demonstrated that in a ten-day experiment the *vertical-aware team-draft interleaving* (VA-TDI) is able to reach a conclusion, while A/B-testing metrics often contradict each other (Section 8.7.4).
- Both vertical-aware and conventional interleaving methods are unbiased under random clicks (Section 8.8.1).

We have shown that vertical-aware interleaving methods can accurately compare result lists while preserving vertical blocks and quality level. Accuracy is as high as for conventional interleaving methods, with only small losses in efficiency for small sample sizes. We also confirmed that methods based on optimized interleaving (OI, VA-OI) usually outperform methods based on team-draft (TDI, VA-TDI). Based on an extensive analysis, we conclude that vertical-aware interleaving methods (VA-TDI, VA-OI) should be used for comparing two ranking systems in situations where vertical documents are present.

One limitation of our work is that not all the experiments were validated in a real search settings. We should note however, that in cases when we did perform experiments with real click log data, we found them to be in line with the findings reported using synthetic or model data. Another limitation is that for unbiasedness (**RQ5.3**) we only considered the case where *the user* clicks randomly, but did not analyse the case where *the ranking systems* are indistinguishable. The problem with that is that it is hard to define what indistinguishable systems are. One may even claim that real-world systems are always different, we just do not always have enough data or the right instruments to observe this.

Future directions. As future work it would be interesting to apply our analysis of the influence on the user experience as presented in Section 8.6 (**RQ5.1**) to the online learning

[19] The code of our simulation experiments, including a reference implementation of the vertical-aware interleaving methods, is publicly available at `https://bitbucket.org/ilps/lerot` as part of the Lerot distribution [137].

to rank problem of *balancing exploration and exploitation* (see, e.g., [81]). Currently, in formulations of the dueling bandits problem for online learning to rank [165], the regret function of an interleaving method—how much the users loose by using a suboptimal system—is set to an average of the relative quality degradation by A and B. As we saw in Section 8.6, the quality of an interleaved list can be far from that average; moreover it depends on the interleaving method used and the ranking systems being interleaved.

Another direction of future work concerns an analysis of A/B-testing methods for aggregated search and their comparison to the interleaving methods presented here. In our work we considered only one very simple online A/B-testing metric, *VertClick*, but one can go further and adapt conventional online metrics to aggregated search settings or introduce new click metrics or new A/B-testing procedures.

8.10 Appendix: Synthetic Ranking Pair Simulation

Fixed position. First we describe how we simulate ranking pairs for organic web documents with vertical documents inserted afterwards (condition *fixed*).

Algorithm 8.5 Generate a pair of organic web rankings.

function GENERATERANKINGPAIR(d, $N_{R,max}$)
  $documentPool \leftarrow$ GENERATEDOCUMENTPOOL$(10 + d, N_{R,max})$
  $A \leftarrow$ GENERATERANKING$(documentPool, 10)$
  $B \leftarrow$ GENERATERANKING$(documentPool, 10)$
  return (A, B)

function GENERATEDOCUMENTPOOL(N', $N_{R,max}$)
  $N_R \leftarrow$ RANDOMINTEGER$(1, N_{R,max})$
  return document list of length N' with N_R relevant documents

function GENERATERANKING(X, N)
  initialize $s_X(d)$ using (8.18)
  $L \leftarrow []$
  while $|L| < N$ **do**
    $d_{\text{next}} \leftarrow$ SAMPLEWITHOUTREPLACEMENT $(s_X(d))$
    $L \leftarrow L + d_{\text{next}}$
  return L

The algorithm for GENERATERANKINGPAIR function (Algorithm 8.5) consists of several steps. In order to generate two ranked lists of size 10 we first generate a document pool of slightly bigger size $(10 + d)$ and then draw documents from this pool to produce rankings A and B. The reason is that rankings A and B can differ not only in the order of the documents, but also one of them can have some additional documents that the other one does not have. In order to allow different document orders in A and B we employ a softmax distribution s_X by Hofmann et al. [80] in which the probability of selecting the next document is inversely proportional to a power of the rank $r_X(d)$ of a document d in

a document pool X:

$$P_{s_X}(d) = \frac{\frac{1}{r_X(d)^\tau}}{\sum_{d' \in X} \frac{1}{r_X(d')^\tau}}. \tag{8.18}$$

In this distribution the document from the bottom of the list X has lower probability of being selected at each step of generating a ranking (Algorithm 8.5, line 13) than one of the top documents. It leads to ranking lists A and B being quite similar to each other (which is usually the case where B is a small experimental ranking change), and we can control the degree of similarity by varying the τ parameter.

For our experiments we set $N_{R,max} = 3$ as in [40]. We also set $d = 2$ and $\tau = 5$ to resemble the level of difference between A and B that we have in the real interleaving experiments.[20]

Algorithm 8.6 Generate a pair of rankings with the fixed vertical blocks $vBlocks$ (condition *fixed*).

1: **function** GENERATERANKINGPAIRWITHVERTICALSFIXED(d, $N_{R,max}$, $vBlocks$, $vPositions$)
2: $(A, B) \leftarrow$ GENERATERANKINGPAIR(d, $N_{R,max}$)
3: insert $vBlocks$ to A at positions $vPositions$
4: insert $vBlocks$ to B at positions $vPositions$
5: **return** (A, B)

If the vertical blocks are *fixed*, then a synthetic ranker is generated in two steps (Algorithm 8.6). First, organic rankings are generated using Algorithm 8.5, then the vertical blocks are inserted at fixed positions. Note, that if the organic list had ten documents, then inserting a vertical block can only increase the total document count, so there will be more than ten documents.

Non-fixed position. Let us now describe the procedure of generating pairs of rankings that contain vertical documents (condition *non-fixed*). We assume that we have a list of vertical types $\{v_t \mid t \in \{1, \dots, T\}\}$ and a probability distribution q, such that for each document d and each vertical type t, document d has type v_t with probability q_t. We also require $\sum_{t=1}^{T} q_t < 1$, so that the probability of the fact that d is a Web (non-vertical) document can be set to $\left(1 - \sum_{t=1}^{T} q_t\right)$:

$$\forall t \quad P(vert(d) = v_t) = q_t \tag{8.19}$$

$$P\left(vert(d) = \text{Web}\right) = 1 - \sum_{t=1}^{T} q_t. \tag{8.20}$$

Now we first generate rankings that have Web and vertical documents mixed up, and then reorder them to respect the vertical blocks. See Algorithm 8.7 for more details.

[20] 97% of the real ranking pairs (Section 8.5.1) has no more than two distinct document pairs, so we choose $d = 2$. The parameter τ is chosen such that the percentage of cases where A and B have 0, 1 or 2 different document pairs resembles that of the real ranking pairs we analyze.

Algorithm 8.7 Generate a pair of rankings with vertical documents (condition *non-fixed*).

function GENERATERANKINGPAIRWITHVERTICALSNONFIXED(d, $N_{R,max}$, q)
  $documentPool \leftarrow$ GENERATEDOCPOOLWITHVERTICALS$(10 + d, N_{R,max}, q)$
  $A \leftarrow$ GENERATERANKINGWITHVERTICALS$(documentPool, 10)$
  $B \leftarrow$ GENERATERANKINGWITHVERTICALS$(documentPool, 10)$
  return (A, B)

function GENERATEDOCPOOLWITHVERTICALS(N', $N_{R,max}$, q)
  $N_R \leftarrow$ RANDOMINTEGER$(1, N_{R,max})$
  $L_{\text{pool}} \leftarrow []$
  while $|L_{\text{pool}}| < N'$ **do**
    $d_{\text{next}} \leftarrow$ document of type t according to the distribution q (8.19)–(8.20)
    $L_{\text{pool}} \leftarrow L_{\text{pool}} + d_{\text{next}}$
  assign exactly N_R documents in L_{pool} to be relevant
  return L_{pool}

function GENERATERANKINGWITHVERTICALS(X, N)
  initialize $s_X(d)$ using (8.18)
  $L \leftarrow []$
  while $|L| < N$ **do**
    $d_{\text{next}} \leftarrow$ SAMPLEWITHOUTREPLACEMENT $(s_X(d))$
    $L \leftarrow L + d_{\text{next}}$
  for each vertical type t **do**
    reorder documents of type t directly below the top document of type t
  return L

9 Conclusions

In this thesis we studied the problem of understanding and modeling users of modern search engines. We identified the main problems arising from the modern complex layout of search engine result pages and suggested solutions to them.

First, we turned our attention to the problem of user modeling in Chapter 4. Namely, we suggested several methods to improve existing click models. We demonstrated that by taking user intent distribution and non-trivial SERP layout into account, we can substantially improve the accuracy of user modeling.

Chapter 5 is dedicated to the question of click model evaluation. We started with a comprehensive comparison of click models along different dimensions. Further, a new evaluation method was introduced to complement a click model evaluation suite that is specifically geared towards complex SERPs.

After studying user models, we turned our attention to the problem of Cranfield-style evaluation metrics, which often lack a connection to user models. In Chapter 6 we introduced a framework to build offline evaluation metrics from click models. We performed a comprehensive evaluation of the new metrics and showed that they are more aligned with online experiments and are more robust in the case of missing judgements.

In Chapter 7 we further improved evaluation metrics by looking at signals other than clicks. We built an evaluation metric that takes explicit and implicit user attention and satisfaction signals into account. We then demonstrated that a user model built in such a way and the corresponding evaluation metric show better alignment with user-reported satisfaction.

Finally, in Chapter 8 we studied an online evaluation method, namely interleaving, and showed how it can be adopted to the settings of complex SERPs in a way that does not affect the user experience.

In Section 9.1 we revisit the research questions formulated in the introduction and summarize our answers to them. In Section 9.2 we discuss some possible directions for future research that naturally follow from this thesis.

9.1 Main Findings

The first research question was the following:

RQ1 Can we substantially improve existing click models by taking into account result page structure and aggregated user characteristics?

Since all commonly used click models ignore the heterogeneous structure of the result page and the spectrum of the user intents, it is clear that there is room for improvement there. We broke down this research question as follows:

RQ1.1 How can we use page layout information to improve click models?

RQ1.2 How can we use aggregated user characteristics such as vertical orientation to improve existing click models?

The very first question **RQ1.1** was about layout, and, first of all, about the presentation type of the results on the page. For instance, news items or image answers are visually different from general web results. In Chapter 4 we suggested a modification to existing click models wherein the parameters of the model that determine the probability of examination were made dependent on the presentation type of the document. We showed that such a modification gives a significant improvement to the model, even when the presentation difference is rather subtle. However, when the presentation difference was removed completely, the gain disappeared, suggesting that this was indeed due to the visual difference and not due to some intrinsic differences between result types. Similarly, we demonstrated that explicitly including pagination buttons into a click model improved its performance.

In **RQ1.2** we asked ourselves whether vertical orientation, or user intent—the aggregated characteristic of the kind of documents the users are interested in for a particular query—could also help to built a better click model. By incorporating intent distribution into the click model and training separate relevance-related parameters for each intent, we showed that it gives an improvement even bigger than the presentation type. Intuitively, by training separate relevance parameters for "jaguar, the car" and "jaguar, the animal" we built a better model of user behavior in response to the query `[jaguar]`, where results answering both intents—to different degrees—are present. We also showed that a combination of the two ideas—layout and intent—gave the biggest improvement.

The next question, **RQ2**, was of a different nature. It concerned the problem of click model evaluation and was answered in Chapter 5.

RQ2 How do we evaluate click models?

We split it into two sub-questions:

RQ2.1 How do different click models perform when evaluated on a common dataset?

RQ2.2 How should we evaluate click models for complex aggregated SERPs?

First of all, we gave an answer to **RQ2.1** in the form of evaluation experiments with an open-source software library and a publicly available dataset. Our experiments were the first to evaluate all commonly used models on the same data and along the same comprehensive set of dimensions.

We then moved to evaluating vertical-aware models. We suggested a new evaluation facet—intuitiveness—wherein models are compared in terms of how well they manage to capture different aspects of the modern aggregated search systems and the complex SERPs they produce. We showed that intuitiveness evaluation gives additional insights into model's quality, thus answering **RQ2.2**.

Having studied click models, we turned our attention to one particular application of them:

RQ3 How can we make use of click models to improve offline evaluation metrics?

In particular, this question was broken down as follows:

RQ3.1 Can we make use of click models to build better evaluation metrics? How do such click model-based IR metrics differ from traditional offline metrics?

RQ3.2 Which evaluation metrics are better tied to the user? Do click model-based metrics show higher agreement with online experiments? How do they compare in terms of discriminative power?

RQ3.3 How well do different offline metrics perform in the presence of unjudged documents?

RQ3.4 How can we modify offline metrics to enhance agreement with online experiments?

To answer this question, in Chapter 6 we developed a framework to convert any click model into evaluation metrics. We considered two variants of metrics: utility-based and effort-based. We then showed that while some of them are quite well correlated with traditional evaluation metrics such as DCG or ERR introduced in Section 3.2, others behave quite differently. This answered **RQ3.1**.

Moreover, while answering **RQ3.2** we showed that our click model-based metrics and, in particular, uUBM (utility-based metric based on UBM) have higher agreement with absolute and pairwise online experiments, while still keeping discriminative power at a reasonable level.

To answer **RQ3.3** we varied the ratio of unrated results in the rating pool and examined how different offline metrics react to it. We found that all offline metrics, both classic and click model-based, degrade when the number of unjudged documents increases. Effort-based metrics based on DBN and DCM were the best at dealing with the unjudged documents.

Finally, to answer **RQ3.4** we suggested two techniques to improve performance. First is *condensation* which simply dictated that we cross out the unjudged documents from the list and thus "condense" the ranked list. Another technique was *thresholding* where we excluded queries with small metric differences when comparing two systems. Both

techniques dramatically improved the metric quality (measured by its correlation with the outcomes of the interleaving experiments) if the metric was based on a user model. The combination of these two techniques yielded the best result.

To continue on the topic of offline evaluation metrics, we asked ourselves the following question:

RQ4 How can we improve user models and offline evaluation metrics to account for non-trivial attention patterns and direct usefulness of result snippets?

The question was split into sub-questions as follows:

RQ4.1 Does a model that unites attention and click signals give more precise estimations of user behavior on a SERP and self-reported satisfaction? How well does the model predict click vs. satisfaction events?

RQ4.2 Does an offline evaluation metric based on such a model show higher agreement with user-reported satisfaction than conventional metrics such as DCG?

We started Chapter 7 by showing that the abandonment rate for answer-seeking queries grows with improvements in the quality of the document snippets, which suggests that users were indeed finding answers in those snippets and were getting satisfied without clicks. To put this into a user model, we suggested the so-called *clicks, attention and satisfaction* (CAS) framework where we added the possibility to gain utility from SERP elements that were both clicked and merely attracted gaze, as well as an explicit user satisfaction model. We showed that such a model achieves reasonable click prediction performance while at the same time being much better at predicting user-reported satisfaction, thereby answering **RQ4.1**.

Finally, in the spirit of **RQ3** we converted that user model into an evaluation metric and showed that the resulting metric was quite different from conventional metrics, while at the same time showing much higher agreement with user-reported satisfaction, which it managed to generalize from the training data. This answered **RQ4.2**.

Finally, the last research question was about a different kind of evaluation, namely online interleaving evaluation based on implicit user feedback:

RQ5 How can we perform accurate online quality evaluation on complex SERPs without affecting the user experience?

In Chapter 8 we suggested a family of algorithms that we called *vertical-aware interleaving*. In parallel, we developed an evaluation framework where an online comparison method was scrutinized from different angles. Therefore, we formulated our sub-questions, each asking about different aspects of the method's quality:

RQ5.1 **Influence on the user experience.** What effect do different interleaving methods—both conventional and newly introduced vertical-aware—have on the user experience in the case of complex SERPs? Do any of these methods run the risk of degrading the quality of the results or altering the user experience?

RQ5.2 **Correctness and sensitivity.** Do different interleaving methods always draw correct conclusions about the better system? How fast, in terms of the number of impressions and amount of feedback needed, can they detect that one aggregated search system is to be preferred over another?

RQ5.3 **Unbiasedness.** Do the interleaving methods that we consider provide a fair and unbiased comparison, or do some of them erroneously infer a preference for one aggregated search system over another in situations where implicit feedback is provided by a randomly clicking user?

To answer **RQ5.1** we showed that vertical-aware interleaving indeed does not degrade nor change the user experience, while the classical methods do run such risk. In response to **RQ5.2** we showed that vertical-aware interleaving methods can find differences between rankings as fast as their conventional counterparts and faster than the A/B-testing method based on absolute metrics. The last one, **RQ5.3** was more of a validation question, which we answered positively for both classical and vertical-aware methods—none of them detected a preference in situations where there was not any.

Above we summarized our main findings and advances towards understanding and modeling users of modern search engines. We demonstrated that modern SERPs are much more complex and often require different machinery when it comes to evaluation or user modeling. Methodologically, we started with existing IR foundations and challenged some of them. We did not completely discard the foundations, but tried to adopt them to the new reality where SERPs have become richer and more interactive. We demonstrated that such an evolutional approach is able to bridge some important gaps and bring us closer to the goal of understanding and modeling the user, provided that we question the existing approaches often enough. A good example of that is what we did when we replaced ad-hoc evaluation metrics by model-based metrics in Chapter 6 or augmented result relevance by snippet relevance in Chapter 7.

Work done in this thesis enables us to do the usual things such as offline/online evaluations and user modeling, knowing that the complex nature of the result pages is taken into account. We also provide general guidelines on how to advance the area even further, which will be the topic of the next section.

9.2 Future Directions

The problem formulated in the introduction—understanding users of modern SERPs and using this understanding for user modeling and quality evaluation—is an open-ended one. We cannot claim that we completely solved it, but rather say how much we advanced in the right direction. We claim that we did advance substantially in accuracy of user modeling and offline evaluation and enabled the use of online interleaving evaluation which was previously not possible in the setting of complex SERPs. Nevertheless, our work has some limitations which motivate future work. While studying the question of user modeling, we limit ourselves to the framework of probabilistic graphical models and did not study other approaches, such as structure learning or neural networks. While studying the question of offline evaluation, we did not address the problem of mismatch between raters' judgements and the users' perception of relevance nor did we test our

CAS model (Chapter 7) on a large-scale data. Finally, in the area of online evaluation, we focused on interleaving methods, while absolute A/B-testing metrics, which have some advantages, were left behind.

There are three different (but related) directions for future work: user modeling, offline evaluation and online evaluation. Below we discuss them in more detail.

9.2.1 User Modeling

The work on user modeling presented in this thesis (Chapters 4 and 5) is by no means complete. There are still ways to improve our understanding of the user. We introduced a number of different signals, such as layout information, user intent and attention, and suggested including them into a user model. All these ideas helped to advance the state of the art of user modeling for complex SERPs. But we have not challenged the underlying fundamentals of these models, which is probabilistic graphical models with a rigid structure defined beforehand.

Another limitation lies in the area of evaluation. While we performed a comprehensive analysis of existing evaluation metrics and suggested a new one in Chapter 5, we do not always know how improvements in these evaluation metrics translate into accuracy of the models in practice.

One possible direction for better user modeling would be learning the structure of the dependency graph using a method described, e.g., in [102, Chapter 18]. The very fact that there are so many different models out there, suggests that we, humans, cannot easily engineer a perfect dependency graph by hand. Instead, we may want to infer this structure from the data, just as we currently do for model parameters. Another way would be replacing the directed Bayesian networks that we normally use by undirected Markov networks, by partially directed conditional random fields or by an altogether different architecture, such as neural networks. In fact some recent work has started to explore the idea of applying neural networks to model search engine users [16, 170]. We believe this to be a promising direction, given that modern SERPs are full of images and rich content, which motivates borrowing techniques from the area of image perception, where neural networks recently helped to achieve extraordinary improvements.

In the area of evaluation, we need to develop application-driven ways to compare models. If an offline evaluation metric based on a user model shows better agreement with the user-reported satisfaction, then it is a better model. If ranking systems winning simulation-based online experiments also win experiments performed on real users, then the model behind the simulation is solid. While it is hard to perform such studies at scale, it is an important part of confirming the validity of our click model evaluation.

9.2.2 Offline Evaluation

It is always hard to evaluate evaluation methods and our work in Chapters 6 and 7 faces the same problem. We did show that our offline metrics correlate with online evaluation signals and user-reported satisfaction, but more evidence from different search engines may be needed before the metrics we suggested will become standard. Since the metrics

themselves requires training, it is still an open question how portable they are between different search systems and how stable they are over time.[1]

As for future work, apart from training on more and different data and improving the underlying user models, we would like to outline two important directions. One direction is a further study of the problem of utility aggregation. While in Chapter 7 we simply assumed utility to be a linear combination of expected snippet-based and document-based relevance gained, the real connection between them may be more involved.

Another direction is the problem of using crowdsourcing for collecting full document relevance and snippet relevance. There are many issues coming from the quality of crowdsourcing annotations: we need a better understanding of how the task should be designed, how we should go about filtering lazy workers and leveraging natural disagreement between the honest ones, to name a few. All of these questions are important in the context of collecting reusable judgements, and our dual relevance nature adds a new dimension to it.

9.2.3 Online Evaluation

There are two limitations of our online evaluation analysis presented in Chapter 8. First, it is the lack of real-world experiments for some of the hypotheses that we tested. In particular, when we studied the effect on the user, we were very risk-averse and opted for performing most of our experiments on simulated users. Also, repeating our real-world experiments in a multi-vertical setting is a limitation that is, perhaps, easier to overcome, provided that whoever does it, has access to the relevant log data. A second limitation is the amount of attention we devoted to the A/B-testing experiments. While interleaving was shown to be more sensitive in early work by [27], A/B-testing is still widely used [64, 67, 101] due to its intuitive nature and the ease of adaptation to heterogeneous SERPs (or even to pages that are not SERPs). Moreover, there are now several techniques helping to improve the sensitivity of A/B-testing experiments [61, 98], which make it an interesting target to study in the context of heterogeneous SERPs.

Apart from mitigating the limitations outlined above, there are several avenues for future work. First, it is an explore-exploit trade-off. How much of a user degradation can we tolerate in order to get a useful signal about the system quality? While the problem of online learning is an active area of research [141, 177], the design of an interleaving method and the regret computation can affect the outcomes and has to be put on the roadmap.

Another direction is improving the accuracy of the experimental outcome. This can be done by adding additional signals, such as mouse movements or viewport position, but also by analysing different ways to interpret the signals to make experimental outcome agree more with user retention metrics. Schuth et al. [140] have made some steps to bridge the gap between interleaving and absolute quality metrics, but to bring SERP-level interleaving signals closer to long-term metrics such as retention, we need to design evaluation methods to account for user interactions spanning multiple queries.

[1] One should note that metric stability, although desirable, should not be put before its accuracy. A good example is DCG, which is widely used and has a fixed and easy-to-compute form, but is quite inaccurate as we demonstrated in Chapter 7.

Acronyms

AP average precision (see page 26)

AS aggregated search (see page 12)

AS_{RBP} aggregated search metric based on rank-biased precision (see page 16)

BI balanced interleaving (see page 17)

BM25 best match 25 (see page 12)

CAS clicks, attention and satisfaction (see page 83)

CCM click chain model (see page 8)

CG cumulative gain (see page 27)

CLEF Conference and Labs of the Evaluation Forum (see page 15)

CM cascade model (see page 21)

CTR click-through rate (see page 16)

DBN dynamic Bayesian network (see page 23)

DCG discounted cumulative gain (see page 26)

DCI document constraints interleaving (see page 17)

DCM dependent click model (see page 24)

DCTR document-based CTR model (see page 20)

EBU expected browsing utility (see page 67)

EB_UBM exploration bias user browsing model (see page 42)

EB_UBM-IA exploration bias user browsing model (intent-aware) (see page 42)

EM expectation maximization (see page 14)

ERR expected reciprocal rank (see page 27)

ERR-IA expected reciprocal rank (intent-aware) (see page 16)

FCM federated click model (see page 56)

IDF inverse document frequency (see page 11)

IR information retrieval (see page 1)

IS item selection (see page 49)

mFCM multi-vertical federated click model (see page 56)

mFCM-NO multi-vertical federated click model (no orientation) (see page 58)

MLE maximum likelihood estimation (see page 14)

NDCG normalized discounted cumulative gain (see page 26)

NTCIR NII (National Institute of Informatics) test collection for information resources (see page 15)

OI optimized interleaving (see page 17)

PBM position-based model (see page 21)

PGM probabilistic graphical model (see page 19)

PI probabilistic interleaving (see page 17)

PLC precision at the lowest click (see page 28)

RBP rank-biased precision (see page 26)

RCM random click model (see page 20)

RCTR rank-based CTR model (see page 20)

RP result presentation (see page 49)

SDBN simplified dynamic Bayesian network (see page 24)

SDBN(P) simplified dynamic Bayesian network (pagination-aware) (see page 32)

SDCM simplified dependent click model (see page 51)

SERP search engine result page (see page 1)

TDI team-draft interleaving (see page 17)

TF term frequency (see page 11)

TREC Text REtrieval Conference (see page 1)

UBM user browsing model (see page 23)

UBM-IA user browsing model (intent-aware) (see page 34)

UBM-intents user browsing model (intents) (see page 40)

UBM-layout user browsing model (vertical layout) (see page 40)

VA-OI vertical-aware optimized interleaving (see page 118)

VA-TDI vertical-aware team-draft interleaving (see page 116)

VD vertical diversity (see page 49)

VS vertical selection (see page 49)

Bibliography

[1] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *WSDM*. ACM, 2009. (Cited on pages 13 and 16.)

[2] K. Ali and C. Chang. On the relationship between click-rate and relevance for search engines. *WIT Trans. on Inform. and Comm. Technol.*, 1, June 2006. (Cited on page 67.)

[3] J. Allan, B. Carterette, J. Aslam, and V. Pavlu. Million Query Track 2007 Overview. Technical report, DTIC Document, 2007. (Cited on page 15.)

[4] O. Alonso and R. Baeza-Yates. Design and implementation of relevance assessments using crowdsourcing. In *ECIR*, pages 153–164. Springer, 2011. (Cited on page 15.)

[5] J. Arguello, F. Diaz, J. Callan, and J.-F. Crespo. Sources of evidence for vertical selection. In *SIGIR*, pages 315–322. ACM, 2009. (Cited on pages 13, 42, 84, 110, and 111.)

[6] J. Arguello, F. Diaz, and J. Paiement. Vertical selection in the presence of unlabeled verticals. In *SIGIR*. ACM, 2010.

[7] J. Arguello, F. Diaz, and J. Callan. Learning to aggregate vertical results into web search results. In *CIKM*, pages 201–210. ACM, 2011. (Cited on pages 13, 84, and 125.)

[8] J. Arguello, F. Diaz, J. Callan, and B. Carterette. A methodology for evaluating aggregated search results. In *ECIR*, pages 141–152. Springer, 2011. (Cited on pages 13, 111, and 125.)

[9] O. Arkhipova and L. Grauer. Evaluating mobile web search performance by taking good abandonment into account. In *SIGIR*, pages 1043–1046. ACM, 2014. (Cited on page 86.)

[10] L. Aroyo and C. Welty. The three sides of CrowdTruth. *Human Computation*, 1(1):31–44, 2014. (Cited on pages 96, 106, and 107.)

[11] J. A. Aslam and E. Yilmaz. Inferring document relevance from incomplete information. In *CIKM*, pages 633–642. ACM, 2007. (Cited on page 97.)

[12] L. Azzopardi, M. de Rijke, and K. Balog. Building simulated queries for known-item topics: an analysis using six european languages. In *SIGIR*, 2007. (Cited on page 15.)

[13] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999. (Cited on page 12.)

[14] N. J. Belkin, R. N. Oddy, and H. M. Brooks. Ask for information retrieval: Part i. background and theory. *Journal of documentation*, 38(2):61–71, 1982. (Cited on page 11.)

[15] R. Berendsen, E. Tsagkias, M. de Rijke, and E. Meij. Generating pseudo test collections for learning to rank scientific articles. In *CLEF*, 2012. (Cited on page 15.)

[16] A. Borisov, I. Markov, M. de Rijke, and P. Serdyukov. A Neural Click Model for Web Search. In *WWW*, pages 531–541, 2016. (Cited on page 156.)

[17] L. Boytsov, D. Novak, Y. Malkov, and N. Eric. Off the Beaten Path: Let's Replace Term-based Retrieval with k-NN Search. In *CIKM*, 2016. (Cited on page 11.)

[18] C. Buckley and E. M. Voorhees. Retrieval evaluation with incomplete information. In *SIGIR*, 2004. (Cited on pages 15, 66, and 80.)

[19] C. Buckley, D. Dimmick, I. Soboroff, and E. Voorhees. Bias and the limits of pooling for large collections. *Information retrieval*, 10(6):491–508, 2007. (Cited on page 15.)

[20] S. Büttcher, C. L. A. Clarke, P. C. K. Yeung, and I. Soboroff. Reliable information retrieval evaluation with incomplete and biased judgements. In *SIGIR*, pages 63–70. ACM, 2007. (Cited on pages 15 and 97.)

[21] B. Carterette. System effectiveness, user models, and user utility: a conceptual framework for investigation. In *SIGIR*, 2011. (Cited on pages 67 and 68.)

[22] B. Carterette and J. Allan. Incremental test collections. In *CIKM*, 2005. (Cited on pages 15 and 80.)

[23] B. Carterette, J. Allan, and R. Sitaraman. Minimal test collections for retrieval evaluation. In *SIGIR*, pages 268–275. ACM, 2006. (Cited on page 15.)

[24] B. Carterette, V. Pavlu, and E. Kanoulas. If I had a million queries. In *ECIR*, pages 288–300. Springer, 2009. (Cited on page 15.)

[25] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *WWW*. ACM, 2009. (Cited on pages 2, 8, 23, 35, 41, 44, 56, 69, 70, and 87.)

[26] O. Chapelle, D. Metzler, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *CIKM*. ACM, 2009. (Cited on pages 4, 14, 16, 27, 67, 69, 70, 74, 78, 86, and 87.)

[27] O. Chapelle, T. Joachims, F. Radlinski, and Y. Yue. Large-scale validation and analysis of interleaved

search evaluation. *TOIS*, 30, Feb. 2012. (Cited on pages 2, 16, 17, 28, 65, 66, 134, and 157.)

[28] D. Chen, W. Chen, H. Wang, Z. Chen, and Q. Yang. Beyond ten blue links: enabling user click modeling in federated web search. In *WSDM*, pages 463–472. ACM, 2012. (Cited on pages 14, 32, 36, 42, 56, 57, 58, 59, 61, 67, 69, 71, and 110.)

[29] X. Chen and H. Min. Improving click model by combining mouse movements with click-through data. In *ICSESS*, pages 183–187. IEEE, 2015. (Cited on page 86.)

[30] L. Chilton and J. Teevan. Addressing people's information needs directly in a web search result page. In *WWW*, 2011. (Cited on pages 5 and 85.)

[31] A. Chouldechova and D. Mease. Differences in search engine evaluations between query owners and non-owners. In *WSDM*, pages 103–112. ACM, 2013. (Cited on page 106.)

[32] A. Chuklin. Effective protocols for low-distance file synchronization. *arXiv preprint arXiv:1102.4712*, 2011. (Cited on page 10.)

[33] A. Chuklin and M. de Rijke. The Anatomy of Relevance. Topical, Snippet and Perceived Relevance in Search Result Evaluation. In *SIGIR'14 Workshop on Gathering Efficient Assessments of Relevance (GEAR'14)*, 2014. (Cited on pages 10 and 93.)

[34] A. Chuklin and M. de Rijke. Deriving vertical orientation from anchor-based assessments. In *The First Workshop on Heterogeneous Information Access at WSDM 2015 (HIA 2015)*, 2015. (Cited on page 10.)

[35] A. Chuklin and M. de Rijke. Incorporating Clicks, Attention and Satisfaction into a Search Engine Result Page Evaluation Model. In *CIKM*, 2016. (Cited on page 9.)

[36] A. Chuklin and A. Lavrentyeva. Adult query classification for web search and recommendation. In *Search and Exploration of X-rated Information (WSDM'13 Workshop Proceedings)*, pages 15–16, 2013. (Cited on page 10.)

[37] A. Chuklin and P. Serdyukov. Good Abandonments in Factoid Queries. In *WWW*, pages 483–484. ACM, 2012. (Cited on pages 2 and 10.)

[38] A. Chuklin and P. Serdyukov. Potential Good Abandonment Prediction. In *WWW*, pages 485–486. ACM, 2012. (Cited on pages 10 and 85.)

[39] A. Chuklin and P. Serdyukov. How Query Extensions Reflect Search Result Abandonments. In *SIGIR*, pages 1087–1088. ACM, 2012. (Cited on pages 10 and 85.)

[40] A. Chuklin, A. Schuth, K. Hofmann, P. Serdyukov, and M. de Rijke. Evaluating Aggregated Search Using Interleaving. In *CIKM*, pages 669–678. ACM, 2013. (Cited on pages 10, 56, 57, 58, 59, 116, 124, 127, 143, and 149.)

[41] A. Chuklin, P. Serdyukov, and M. de Rijke. Using Intent Information to Model User Behavior in Diversified Search. In *ECIR*, pages 1–13. Springer, 2013. (Cited on pages 9, 13, 67, and 69.)

[42] A. Chuklin, P. Serdyukov, and M. de Rijke. Click Model-based Information Retrieval Metrics. In *SIGIR*, pages 493–502. ACM, 2013. (Cited on pages 2, 9, 99, and 113.)

[43] A. Chuklin, P. Serdyukov, and M. de Rijke. Modeling Clicks Beyond the First Result Page. In *CIKM*, pages 1217–1220. ACM, 2013. (Cited on page 9.)

[44] A. Chuklin, K. Zhou, A. Schuth, F. Sietsma, and M. de Rijke. Evaluating Intuitiveness of Vertical-aware Click Models. In *SIGIR*, pages 1075–1078. ACM, 2014. (Cited on page 9.)

[45] A. Chuklin, I. Markov, and M. de Rijke. *Click Models for Web Search*. Morgan & Claypool, 2015. (Cited on pages 3, 8, 9, 14, 20, 36, and 43.)

[46] A. Chuklin, I. Markov, and M. de Rijke. An Introduction to Click Models for Web Search: SIGIR 2015 Tutorial. In *SIGIR*, pages 1113–1115, 2015. (Cited on page 10.)

[47] A. Chuklin, I. Markov, and M. de Rijke. Advanced Click Models and Their Applications to IR: SIGIR 2015 Tutorial. In *SIGIR*, pages 1111–1112, 2015. (Cited on page 10.)

[48] A. Chuklin, A. Schuth, K. Zhou, and M. de Rijke. A Comparative Analysis of Interleaving Methods for Aggregated Search. *TOIS*, 33(2), 2015. (Cited on pages 2 and 10.)

[49] A. Chuklin, I. Markov, and M. de Rijke. Click Models for Web Search and Their Applications to IR: WSDM 2016 Tutorial. In *WSDM*, pages 689–690, 2016. (Cited on page 10.)

[50] A. Chuklin, I. Markov, and M. de Rijke. Click Models for Web Search: RuSSIR 2016 Course. In *RuSSIR*, 2016. (Cited on page 10.)

[51] C. Clarke, E. Agichtein, S. Dumais, and R. White. The influence of caption features on clickthrough patterns in web search. In *SIGIR*. ACM, 2007. (Cited on pages 3 and 5.)

[52] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon.

Novelty and diversity in information retrieval evaluation. In *SIGIR*, pages 659–666. ACM, 2008. (Cited on pages 16, 42, and 112.)

[53] C. L. A. Clarke, N. Craswell, and I. Soboroff. A comparative analysis of cascade measures for novelty and diversity. In *WSDM*, pages 75–84. ACM, 2011. (Cited on pages 16, 30, and 81.)

[54] C. L. A. Clarke, N. Craswell, I. Soboroff, and E. M. Voorhees. Overview of the TREC 2011 Web Track. In *TREC*. NIST, 2011. (Cited on pages 68 and 80.)

[55] C. Cleverdon. The cranfield tests on index language devices. In *Aslib proceedings*, volume 19, pages 173–194. MCB UP Ltd, 1967. (Cited on pages 12 and 14.)

[56] C. W. Cleverdon. Report on the testing and analysis of an investigation into the comparative efficiency of indexing systems. Technical report, ASLIB Cranfield project, 1962.

[57] C. W. Cleverdon, J. Mills, and M. Keen. Factors determining the performance of indexing systems. Technical report, ASLIB Cranfield project, 1966. (Cited on pages 1, 12, 14, and 66.)

[58] J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, XX(1):37–46, 1960. (Cited on page 106.)

[59] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM*, page 87. ACM, 2008. (Cited on pages 13, 14, 21, 22, 48, and 69.)

[60] T. Demeester, D. Trieschnigg, D. Nguen, and D. Hiemstra. Overview of the TREC 2013 Federated Web Search track. In *TREC*. NIST, 2013. (Cited on pages 12, 121, 124, 125, and 133.)

[61] A. Deng, Y. Xu, R. Kohavi, and T. Walker. Improving the sensitivity of online controlled experiments by utilizing pre-experiment data. In *WSDM*, pages 123–132. ACM, 2013. (Cited on pages 16 and 157.)

[62] F. Diaz, R. W. White, G. Buscher, and D. Liebling. Robust models of mouse movement on dynamic web search results pages. In *CIKM*. ACM, 2013. (Cited on pages 13, 84, 86, and 90.)

[63] A. Diriye, R. W. White, G. Buscher, and S. T. Dumais. Leaving so soon? Understanding and predicting search abandonment rationales. In *CIKM*, pages 1025–1034. ACM, 2012. (Cited on pages 85, 86, and 94.)

[64] P. Dmitriev and X. Wu. Measuring Metrics. In *CIKM*, pages 429–437, 2016. (Cited on page 157.)

[65] A. Dong, Y. Chang, Z. Zheng, G. Mishne, J. Bai, R. Zhang, K. Buchner, C. Liao, and F. Diaz. Towards recency ranking in web search. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 11–20. ACM, 2010. (Cited on page 122.)

[66] Z. Dou, R. Song, and J.-R. Wen. A large-scale evaluation and analysis of personalized search strategies. In *WWW*, pages 581–590. ACM, 2007. (Cited on page 14.)

[67] A. Drutsa and P. Serdyukov. Future User Engagement Prediction and Its Application to Improve the Sensitivity of Online Experiments. In *WWW*, pages 256–266, 2015. (Cited on pages 16 and 157.)

[68] S. Dumais, E. Cutrell, and H. Chen. Optimizing search by showing results in context. In *CHI*, pages 277–284. ACM, 2001. (Cited on pages 12, 43, 84, and 110.)

[69] G. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *SIGIR*. ACM, 2008. (Cited on pages 2, 14, 23, 32, 35, 38, 42, 48, 67, 69, 71, and 72.)

[70] G. Dupret, V. Murdock, and B. Piwowarski. Web search engine evaluation using clickthrough data and a user model. In *WWW*. ACM, 2007. (Cited on page 14.)

[71] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap (Chapman & Hall/CRC Monographs on Statistics & Applied Probability)*. Chapman and Hall/CRC, 1 edition, May 1994. (Cited on pages 36, 39, 51, and 77.)

[72] R. A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, pages 309–368, 1922. (Cited on page 47.)

[73] A. Grotov, A. Chuklin, I. Markov, L. Stout, F. Xumara, and M. de Rijke. A Comparative Study of Click Models for Web Search. In *CLEF*, volume 9283, pages 78–90. Springer, 2015. (Cited on pages 10 and 61.)

[74] F. Guo, C. Liu, A. Kannan, T. Minka, M. Taylor, Y.-M. Wang, and C. Faloutsos. Click chain model in web search. In *WWW*. ACM, 2009. (Cited on pages 8, 14, 24, 36, and 67.)

[75] F. Guo, C. Liu, and Y. Wang. Efficient multiple-click models in web search. In *WSDM*. ACM, 2009. (Cited on pages 41, 48, 52, 67, and 69.)

[76] Q. Guo and E. Agichtein. Exploring mouse movements for inferring query intent. In *SIGIR*, pages 707–708. ACM, 2008. (Cited on page 94.)

[77] Q. Guo and E. Agichtein. Towards predicting web searcher gaze position from mouse movements. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems*, pages 3601–3606. ACM, 2010. (Cited on page 90.)

[78] Q. Guo, H. Jin, D. Lagun, S. Yuan, and E. Agichtein. Mining touch interaction data on mobile devices to predict web search result relevance. In *SIGIR*, pages 153–162. ACM, 2013. (Cited on page 106.)

[79] J. He, C. Zhai, and X. Li. Evaluation of methods for relative comparison of retrieval systems based on clickthroughs. In *CIKM*, pages 2029–2032. ACM, 2009. (Cited on pages 17, 112, and 132.)

[80] K. Hofmann, S. Whiteson, and M. de Rijke. A probabilistic method for inferring preferences from clicks. In *CIKM*, pages 249–258. ACM, 2011. (Cited on pages 17, 66, 113, 136, and 148.)

[81] K. Hofmann, S. Whiteson, and M. de Rijke. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval*, 16(1):63–90, Apr. 2012. (Cited on page 148.)

[82] K. Hofmann, A. Schuth, S. Whiteson, and M. de Rijke. Reusing historical interaction data for faster online learning to rank for IR. In *WSDM*, pages 183–192. ACM, 2013. (Cited on pages 2, 17, 112, 127, and 139.)

[83] K. Hofmann, S. Whiteson, and M. de Rijke. Fidelity, Soundness, and Efficiency of Interleaved Comparison Methods. *TOIS*, 2013. (Cited on pages 17, 136, 138, and 144.)

[84] K. Hofmann, L. Li, F. Radlinski, and Others. Online Evaluation for Information Retrieval. *Foundations and Trends in Information Retrieval*, 10(1):1–117, 2016. (Cited on pages 5 and 16.)

[85] B. Hu, Y. Zhang, W. Chen, G. Wang, and Q. Yang. Characterizing search intent diversity into click models. In *WWW*. ACM, 2011. (Cited on pages 34 and 42.)

[86] J. Huang and A. Diriye. Web user interaction mining from touch-enabled mobile devices. In *HCIR*, 2012. (Cited on page 106.)

[87] J. Huang, R. W. White, and S. Dumais. No clicks, no problem: using cursor movements to understand and improve search. In *CHI*, pages 1225–1234. ACM, 2011. (Cited on page 86.)

[88] J. Huang, R. W. White, G. Buscher, and K. Wang. Improving searcher models using mouse cursor activity. In *SIGIR*. ACM, 2012. (Cited on pages 3, 14, 32, 67, and 86.)

[89] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *TOIS*, 20(4):422–446, 2002. (Cited on pages 4, 26, 67, 68, and 132.)

[90] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142. ACM, 2002. (Cited on pages 17, 66, 73, 110, and 111.)

[91] T. Joachims. Evaluating retrieval performance using clickthrough data. *Text Mining*, 2003. (Cited on pages 16, 17, 110, and 113.)

[92] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR*. ACM, 2005. (Cited on pages 13, 19, 21, 59, and 61.)

[93] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *TOIS*, 25(2):7, 2007. (Cited on page 16.)

[94] E. Kanoulas. A short survey on online and offline methods for search quality evaluation. In *Information Retrieval*, pages 38–87. Springer, 2016. (Cited on page 16.)

[95] G. Kazai and I. Zitouni. Quality management in crowdsourcing using gold judges behavior. In *WSDM*, pages 267–276. ACM, 2016. (Cited on page 106.)

[96] T. Kenter, K. Balog, and M. de Rijke. Evaluating document filtering systems over time. *Information Processing & Management*, 51(6):791–808, 2015. (Cited on page 122.)

[97] E. Kharitonov, C. Macdonald, P. Serdyukov, and I. Ounis. Optimised scheduling of online experiments. In *SIGIR*, pages 453–462. ACM, 2015. (Cited on page 16.)

[98] E. Kharitonov, A. Vorobev, C. Macdonald, P. Serdyukov, and I. Ounis. Sequential Testing for Early Stopping of Online Experiments. In *SIGIR*, pages 473–482, 2015. (Cited on pages 16 and 157.)

[99] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne. Controlled experiments on the web: survey and practical guide. *Data Mining and Knowledge Discovery*, 18(1):140–181, jul 2008. (Cited on pages 2, 16, and 65.)

[100] R. Kohavi, A. Deng, B. Frasca, R. Longbotham, T. Walker, and Y. Xu. Trustworthy Online Controlled Experiments: Five Puzzling Outcomes Explained. In *KDD*, pages 786–794, 2012. (Cited on page 16.)

[101] R. Kohavi, A. Deng, R. Longbotham, and Y. Xu. Seven rules of thumb for web site experimenters. In

KDD, pages 1857–1866. ACM, 2014. (Cited on pages 16 and 157.)

[102] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Aug. 2009. (Cited on pages 19 and 156.)

[103] K. Krippendorff. Estimating the reliability, systematic error and random error of interval data. *Educational and Psychological Measurement*, 30(1):61–70, 1970. (Cited on page 106.)

[104] J. Li, S. Huffman, and A. Tokuda. Good abandonment in mobile and PC internet search. In *SIGIR*. ACM, 2009. (Cited on pages 2, 5, 85, 86, and 96.)

[105] A. Lipani, M. Lupu, E. Kanoulas, and A. Hanbury. The solitude of relevant documents in the pool. In *CIKM*, pages 1989–1992. ACM, 2016. (Cited on page 15.)

[106] C. Liu, F. Guo, and C. Faloutsos. BBM: bayesian browsing model from petabyte-scale data. In *KDD*. ACM, 2009. (Cited on pages 14 and 67.)

[107] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989. (Cited on page 93.)

[108] Y. Liu, C. Wang, K. Zhou, J. Nie, M. Zhang, and S. Ma. From skimming to reading: A two-stage examination model for web search. In *CIKM*, pages 849–858. ACM, 2014. (Cited on pages 86 and 105.)

[109] Y. Liu, Y. Chen, J. Tang, J. Sun, M. Zhang, S. Ma, and X. Zhu. Different users, different opinions: Predicting search satisfaction with mouse movement information. In *SIGIR*, pages 493–502. ACM, 2015. (Cited on pages 86 and 106.)

[110] H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317, 1957. (Cited on page 11.)

[111] C. Manning, P. Raghavan, and H. Schütze. *An introduction to information retrieval*. Cambridge University Press, New York, NY, USA, 2008. (Cited on page 12.)

[112] I. Markov, E. Kharitonov, V. Nikulin, P. Serdyukov, M. de Rijke, and F. Crestani. Vertical-aware click model-based effectiveness metrics. In *CIKM*, pages 1867–1870, 2014. (Cited on page 2.)

[113] I. Markov, A. Chuklin, and Y. Liu. Modeling Search Behavior in Heterogeneous Environments. In *The First Workshop on Heterogeneous Information Access at WSDM 2015 (HIA 2015)*, pages 6–8, 2015. (Cited on page 10.)

[114] T. P. Minka. A comparison of numerical optimizers for logistic regression. *Unpublished draft*, 2003. (Cited on page 93.)

[115] A. Moffat and J. Zobel. Rank-biased precision for measurement of retrieval effectiveness. *TOIS*, 27(1): 1–27, Dec. 2008. (Cited on pages 14, 21, 26, and 41.)

[116] C. S. Mooers. Coding, information retrieval, and the rapid selector. *Journal of the American Society for Information Science*, 1(4):225, 1950. (Cited on pages 1 and 11.)

[117] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2013. (Cited on page 48.)

[118] V. Navalpakkam and E. F. Churchill. Mouse tracking: Measuring and predicting users' experience of web-based content. In *CHI*, pages 2963–2972. ACM, 2012. (Cited on page 86.)

[119] V. Navalpakkam, L. Jentzsch, R. Sayres, S. Ravi, A. Ahmed, and A. Smola. Measurement and modeling of eye-mouse behavior in the presence of nonlinear page layouts. In *WWW*, pages 953–964. ACM, 2013. (Cited on pages 86 and 90.)

[120] D. Nguyen, T. Demeester, D. Trieschnigg, and D. Hiemstra. Federated search in the wild: the combined power of over a hundred search engines. In *CIKM*, pages 1874–1878. ACM, 2012. (Cited on pages 12, 59, and 125.)

[121] A. K. Ponnuswami, K. Pattabiraman, D. Brand, and T. Kanungo. Model characterization curves for federated search using click-logs: predicting user engagement metrics for the span of feasible operating points. In *WWW*, pages 67–76. ACM, 2011. (Cited on page 13.)

[122] A. K. Ponnuswami, K. Pattabiraman, Q. Wu, R. Gilad-Bachrach, and T. Kanungo. On composition of a federated web search result page: using online users to provide pairwise preference for heterogeneous verticals. In *WSDM*, pages 715–724. ACM, 2011. (Cited on pages 13, 84, and 111.)

[123] F. Radlinski and N. Craswell. Comparing the sensitivity of information retrieval metrics. In *SIGIR*, pages 667–674. ACM, 2010. (Cited on pages 5, 66, 67, 73, 74, 112, and 113.)

[124] F. Radlinski and N. Craswell. Optimized interleaving for online retrieval evaluation. In *WSDM*, pages 245–254. ACM, 2013. (Cited on pages 17, 112, 113, 114, 115, 119, 125, 126, and 142.)

[125] F. Radlinski and S. Dumais. Improving personalized web search using result diversification. In *SIGIR*. ACM, 2006. (Cited on page 12.)

[126] F. Radlinski and T. Joachims. Minimally invasive randomization for collecting unbiased preferences from clickthrough logs. In *Proceedings of the National Conference on Artificial Intelligence*, page 1406. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006. (Cited on page 17.)

[127] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *CIKM*, pages 43–52. ACM, 2008. (Cited on pages 16, 17, 27, 66, 73, 80, 110, 113, 116, and 143.)

[128] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. *NIST Special Publication*, 109:109, 1995. (Cited on page 12.)

[129] K. Rodden, X. Fu, A. Aula, and I. Spiro. Eye-mouse coordination patterns on web search results pages. In *CHI*, pages 2997–3002, 2008. (Cited on pages 85 and 90.)

[130] T. Sakai. Evaluating evaluation metrics based on the bootstrap. In *SIGIR*, 2006. (Cited on pages 67 and 80.)

[131] T. Sakai. Alternatives to Bpref. In *SIGIR*. ACM, 2007. (Cited on pages 5, 15, 67, and 74.)

[132] T. Sakai. Evaluation with informational and navigational intents. In *WWW*, 2012. (Cited on page 49.)

[133] T. Sakai and R. Song. Evaluating diversified search results using per-intent graded relevance. In *SIGIR*. ACM, 2011. (Cited on page 16.)

[134] M. Sanderson and W. B. Croft. The history of information retrieval research. *Proceedings of the IEEE*, 100(Special Centennial Issue):1444–1451, 2012. (Cited on page 12.)

[135] M. Sanderson and H. Joho. Forming test collections with no system pooling. In *SIGIR*. ACM, 2004. (Cited on pages 15 and 80.)

[136] D. Savenkov, P. Braslavski, and M. Lebedev. Search Snippet Evaluation at Yandex : Lessons Learned and Future Directions. In *CLEF*, 2011. (Cited on page 12.)

[137] A. Schuth, K. Hofmann, S. Whiteson, and M. de Rijke. Lerot: an online learning to rank framework. In *Proceedings of the 2013 workshop on Living labs for information retrieval evaluation*. ACM, 2013. (Cited on pages 8, 50, and 147.)

[138] A. Schuth, F. Sietsma, S. Whiteson, D. Lefortier, and M. de Rijke. Multileaved comparisons for fast online evaluation. In *CIKM*. ACM, 2014. (Cited on pages 17 and 113.)

[139] A. Schuth, R.-J. Bruintjes, F. Büttner, J. van Doorn, C. Groenland, H. Oosterhuis, C.-N. Tran, B. Veeling, J. van der Velde, R. Wechsler, et al. Probabilistic multileave for online retrieval evaluation. In *SIGIR*. ACM, 2015. (Cited on page 17.)

[140] A. Schuth, K. Hofmann, and F. Radlinski. Predicting search satisfaction metrics with interleaved comparisons. In *SIGIR*. ACM, 2015. (Cited on page 157.)

[141] A. Schuth, H. Oosterhuis, S. Whiteson, and M. de Rijke. Multileave gradient descent for fast online learning to rank. In *WSDM*. ACM, 2016. (Cited on pages 17 and 157.)

[142] J. Seo, W. B. Croft, K. H. Kim, and J. H. Lee. Smoothing click counts for aggregated vertical search. In *Advances in Information Retrieval*, pages 387–398. Springer, 2011. (Cited on pages 13 and 43.)

[143] S. Shen, B. Hu, W. Chen, and Q. Yang. Personalized click model through collaborative filtering. In *WSDM*, New York, New York, USA, 2012. ACM. (Cited on page 14.)

[144] L. Si and J. Callan. Modeling search engine effectiveness for federated search. In *SIGIR*, pages 83–90. ACM, 2005. (Cited on page 12.)

[145] C. Silverstein, M. Henzinger, H. Marais, and M. Moricz. Analysis of a very large altavista query log. Technical report, Systems Research Center, Compaq Computer Corporation, 1998. (Cited on page 84.)

[146] M. D. Smucker and C. L. A. Clarke. Time-based Calibration of Effectiveness Measures. In *SIGIR*. ACM, 2012. (Cited on page 81.)

[147] Y. Song, X. Shi, R. White, and A. Hassan. Context-Aware Web Search Abandonment Prediction. In *SIGIR*, 2014. (Cited on page 85.)

[148] K. Spärck Jones. Report on the need for and provision of an "ideal" information retrieval test collection. Technical report, British Library Research and Development Report 5266, Computer Laboratory, University of Cambridge, 1975. (Cited on page 15.)

[149] K. Spärck Jones. *Readings in information retrieval*. Morgan Kaufmann, 1997. (Cited on page 12.)

[150] K. Spärck Jones and C. van Rijsbergen. Information retrieval test collection. *J. documentation*, 32(1): 59–75, 1976. (Cited on page 12.)

[151] S. Stamou and E. N. Efthimiadis. Interpreting user inactivity on search results. In *ECIR*, pages 100–113. Springer, 2010. (Cited on page 84.)

[152] A. Styskin. Aggregate and conquer: Finding the way in the diverse world of user intents. In *Industry Day, ECIR*. Springer, 2013. (Cited on pages 13 and 110.)

[153] A. Styskin, F. Romanenko, F. Vorobyev, and P. Serdyukov. Recency ranking by diversification of result set. In *CIKM*, pages 1949–1952. ACM, 2011. (Cited on page 13.)

[154] S. Sushmita, H. Joho, M. Lalmas, and R. Villa. Factors affecting click-through behavior in aggregated search interfaces. In *CIKM*, pages 519–528. ACM, 2010. (Cited on pages 3, 5, 59, 61, and 125.)

[155] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *SIGIR*, pages 449–456. ACM, 2005. (Cited on page 14.)

[156] A. Tombros and M. Sanderson. Advantages of query biased summaries in information retrieval. In *SIGIR*, pages 2–10. ACM, 1998. (Cited on page 12.)

[157] A. Turpin, F. Scholer, K. Jarvelin, M. Wu, and J. S. Culpepper. Including summaries in system evaluation. In *SIGIR*. ACM, 2009. (Cited on pages 21, 71, 80, and 85.)

[158] E. M. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. *Information processing & management*, 36, 2000. (Cited on page 80.)

[159] C. Wang, Y. Liu, M. Zhang, S. Ma, M. Zheng, J. Qian, and K. Zhang. Incorporating vertical results into search click models. In *SIGIR*, pages 503–512. ACM, 2013. (Cited on pages 14, 42, and 84.)

[160] C. Wang, Y. Liu, M. Wang, K. Zhou, J. Nie, and S. Ma. Incorporating non-sequential behavior into click models. In *SIGIR*, pages 283–292. ACM, 2015. (Cited on pages 8 and 84.)

[161] K. Williams, J. Kiseleva, A. C. Crook, I. Zitouni, A. H. Awadallah, and M. Khabsa. Detecting Good Abandonment in Mobile Search. In *WWW*, pages 495–505, 2016. (Cited on pages 2 and 85.)

[162] I. H. Witten, A. Moffat, and T. C. Bell. *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann, 1999. (Cited on page 11.)

[163] Q. Xing, Y. Liu, J.-Y. Nie, M. Zhang, S. Ma, and K. Zhang. Incorporating user preferences into click models. In *CIKM*, pages 1301–1310. ACM, 2013. (Cited on page 14.)

[164] E. Yilmaz, M. Shokouhi, N. Craswell, and S. Robertson. Expected browsing utility for web search evaluation. In *CIKM*. ACM, 2010. (Cited on pages 67, 68, 69, and 70.)

[165] Y. Yue and T. Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *ICML*, pages 1201–1208. ACM, 2009. (Cited on page 148.)

[166] Y. Yue, Y. Gao, O. Chapelle, Y. Zhang, and T. Joachims. Learning more powerful test statistics for click-based retrieval evaluation. In *SIGIR*, 2010. (Cited on page 67.)

[167] Y. Zhang, L. A. F. Park, and A. Moffat. Click-based evidence for decaying weight distributions in search effectiveness metrics. *Information Retrieval*, 13(1), 2010. (Cited on pages 21 and 59.)

[168] Y. Zhang, D. Wang, G. Wang, W. Chen, Z. Zhang, B. Hu, and L. Zhang. Learning click models via probit bayesian inference. In *CIKM*, New York, New York, USA, 2010. ACM. (Cited on page 14.)

[169] Y. Zhang, W. Chen, D. Wang, and Q. Yang. User-click modeling for understanding and predicting search-behavior. In *KDD*. ACM, 2011. (Cited on pages 14, 36, 67, and 69.)

[170] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T.-Y. Liu. Sequential click prediction for sponsored search with recurrent neural networks. In *AAAI*, pages 1369–1375, 2014. (Cited on page 156.)

[171] K. Zhou, R. Cummins, M. Lalmas, and J. M. Jose. Evaluating aggregated search pages. In *SIGIR*, pages 115–124. ACM, 2012. (Cited on pages 13, 16, 58, 84, 111, 112, and 132.)

[172] K. Zhou, R. Cummins, M. Lalmas, and J. M. Jose. Evaluating reward and risk for vertical selection. In *CIKM*, pages 2631–2634. ACM, 2012. (Cited on page 111.)

[173] K. Zhou, R. Cummins, M. Lalmas, and J. M. Jose. Which vertical search engines are relevant? In *WWW*, pages 1557–1568. ACM, 2013. (Cited on page 111.)

[174] K. Zhou, M. Lalmas, T. Sakai, R. Cummins, and J. M. Jose. On the reliability and intuitiveness of aggregated search metrics. In *CIKM*, pages 689–698. ACM, 2013. (Cited on pages 49, 50, 58, 59, 112, 121, 123, 124, 125, and 132.)

[175] K. Zhou, T. Demeester, D. Nguyen, D. Hiemstra, and D. Trieschnigg. Aligning vertical collection relevance with user intent. In *CIKM*. ACM, 2014. (Cited on page 111.)

[176] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM computing surveys (CSUR)*, 38(2):6, 2006. (Cited on page 11.)

[177] M. Zoghi, S. Whiteson, and M. de Rijke. MergeRUCB: A method for large-scale online ranker evaluation. In *WSDM*, pages 17–26. ACM, 2015. (Cited on page 157.)

Summary

Understanding and Modeling Users of Modern Search Engines

As search is being used by billions of people, modern search engines are becoming more and more complex. And complexity does not just come from the algorithms. Richer and richer content is being added to search engine result pages: news and sports results, definitions and translations, images and videos. Many such elements are added by search engines in their attempt to stand out from the competition by providing a superior user experience. However, the more complex search engines become, the harder it gets to understand users and their interactions with result pages, and to measure the quality of the user experience. In this thesis we address exactly this topic.

We start by analyzing user behavior on complex result pages and show that the users' click patterns are non-trivial. We also demonstrate that there are situations where we observe no clicks at all, even though there is good content on a search engine result page and the users are likely to be satisfied. Having made these observations, we proceed to *modeling* the users of modern search engines. In particular, we suggest so-called *intent-aware* models that capture different user intents and different presentation types on the search engine result page. We also contribute to the area of user model evaluation by presenting a comprehensive evaluation of click models and a new evaluation method designed specifically for modern search engine result pages.

We then turn our attention to *evaluating* the user search experience. With additional tools made available by modern search engines, users hope to become more productive in their everyday life and achieve more in a smaller amount of time. That can reliably be achieved only if the added complexity of the search engines comes with a correctly measured benefit. We suggest a general framework for deriving an offline evaluation metric from a click model—*click model-based metrics*—and then refine the underlying user model by modeling user attention and user satisfaction signals in addition to clicks. The benefit of such evaluation metrics is that they can be fit to real user data and then re-used multiple times without involving actual users.

Next to studying offline metrics, we also study online evaluation approaches that require deploying new versions of the search system live and comparing them to the current production system based on user feedback. We propose a modification of a popular online evaluation method—interleaving—that accounts for blocks of rich content (often called verticals) that we refer to as *vertical-aware interleaving*. In addition, we suggest several methods for evaluating evaluation methods, both offline and online ones, from multiple angles.

Samenvatting

Modelleren en Begrijpen van Gebruikers van Moderne Zoekmachines

Nu zoekmachines door miljarden mensen worden gebruikt, neemt hun complexiteit toe. De complexiteit zit niet alleen in de achterliggende algoritmes, maar ook in de resultaatpagina's, die steeds rijkere resultaten tonen, zoals nieuws, sport, vertalingen, afbeeldingen en video's. Meer en meer van deze elementen worden toegevoegd door zoekmachines, die hun concurrenten af willen troeven in het bieden van de beste gebruikerservaring. Echter, hoe complexer de zoekmachines worden, hoe moeilijker het wordt om de interactie van gebruikers met de zoekmachine te interpreteren en om de kwaliteit van de gebruikerservaring te meten. Dit proefschrift behandelt precies dit onderwerp.

We beginnen met het analyseren van het gedrag van gebruikers op complexe resultaatpagina's en tonen aan dat hun klikgedrag niet-triviaal is. Bovendien laten we zien dat er situaties zijn waarin geen enkele klik wordt geobserveerd, terwijl er wel relevante informatie getoond wordt en de gebruiker waarschijnlijk tevreden is. Op basis van deze bevindingen *modelleren* we de gebruikers van moderne zoekmachines. We stellen *intent-aware* modellen voor die verschillende gebruikersdoelen en verschillende presentatietypes op de resultaatpagina modelleren. We dragen ook bij aan de evaluatie van gebruikersmodellen met een uitgebreide evaluatie van klikmodellen, en de introductie van een nieuwe evaluatiemethode die specifiek gericht is op resultaatpagina's van moderne zoekmachines.

Vervolgens richten we onze aandacht op het evalueren van de gebruikerservaring. Door alle toevoegingen die moderne zoekmachines bieden, worden gebruikers hopelijk productiever in hun dagelijks leven en kunnen ze meer gedaan krijgen in minder tijd. Een correcte manier van het meten van de toegevoegde waarde van de additionele complexiteit is hiermee zeer gewenst. We stellen een algemene methode voor om een offline evaluatiemaat af te leiden van een klikmodel — click model-based metrics — waarna het onderliggende gebruikersmodel bijgesteld wordt door, naast de kliks, de aandacht van de gebruiker en de afgegeven signalen van tevredenheid te modelleren. Het voordeel van deze evaluatiematen is dat ze geoptimaliseerd kunnen worden op basis van bestaande gebruikersgegevens, en vervolgens opnieuw gebruikt kunnen worden zonder dat er nogmaals gebruikers bij betrokken hoeven te worden.

Naast offline evaluatiematen bestuderen we ook zogenaamde online aanpakken voor evaluatie die uitgaan van meerdere live versies van zoekmachines, die vergeleken worden op basis van actuele gebruikersinteracties. We stellen een aanpassing voor van een veelgebruikte online evaluatiemethode — interleaving — die rekening houdt met additionele elementen in de resultaatpagina (ook wel *verticals* genoemd): *vertical-aware interleaving*. Bovendien stellen we meerdere manieren voor om evaluatiemethodes te evalueren uit verschillende invalshoeken.

SIKS Dissertation Series

1998

1 Johan van den Akker (CWI) *DEGAS: An Active, Temporal Database of Autonomous Objects*
2 Floris Wiesman (UM) *Information Retrieval by Graphically Browsing Meta-Information*
3 Ans Steuten (TUD) *A Contribution to the Linguistic Analysis of Business Conversations*
4 Dennis Breuker (UM) *Memory versus Search in Games*
5 E. W. Oskamp (RUL) *Computerondersteuning bij Straftoemeting*

1999

1 Mark Sloof (VUA) *Physiology of Quality Change Modelling: Automated modelling of*
2 Rob Potharst (EUR) *Classification using decision trees and neural nets*
3 Don Beal (UM) *The Nature of Minimax Search*
4 Jacques Penders (UM) *The practical Art of Moving Physical Objects*
5 Aldo de Moor (KUB) *Empowering Communities: A Method for the Legitimate User-Driven*
6 Niek J. E. Wijngaards (VUA) *Re-design of compositional systems*
7 David Spelt (UT) *Verification support for object database design*
8 Jacques H. J. Lenting (UM) *Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism*

2000

1 Frank Niessink (VUA) *Perspectives on Improving Software Maintenance*
2 Koen Holtman (TUe) *Prototyping of CMS Storage Management*
3 Carolien M. T. Metselaar (UvA) *Sociaal-organisatorische gevolgen van kennistechnologie*
4 Geert de Haan (VUA) *ETAG, A Formal Model of Competence Knowledge for User Interface*
5 Ruud van der Pol (UM) *Knowledge-based Query Formulation in Information Retrieval*
6 Rogier van Eijk (UU) *Programming Languages for Agent Communication*
7 Niels Peek (UU) *Decision-theoretic Planning of Clinical Patient Management*
8 Veerle Coupé (EUR) *Sensitivity Analyis of Decision-Theoretic Networks*
9 Florian Waas (CWI) *Principles of Probabilistic Query Optimization*
10 Niels Nes (CWI) *Image Database Management System Design Considerations, Algorithms and Architecture*
11 Jonas Karlsson (CWI) *Scalable Distributed Data Structures for Database Management*

2001

1 Silja Renooij (UU) *Qualitative Approaches to Quantifying Probabilistic Networks*
2 Koen Hindriks (UU) *Agent Programming Languages: Programming with Mental Models*
3 Maarten van Someren (UvA) *Learning as problem solving*
4 Evgueni Smirnov (UM) *Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets*
5 Jacco van Ossenbruggen (VUA) *Processing Structured Hypermedia: A Matter of Style*
6 Martijn van Welie (VUA) *Task-based User Interface Design*
7 Bastiaan Schonhage (VUA) *Diva: Architectural Perspectives on Information Visualization*
8 Pascal van Eck (VUA) *A Compositional Semantic Structure for Multi-Agent Systems Dynamics*
9 Pieter Jan 't Hoen (RUL) *Towards Distributed Development of Large Object-Oriented Models*
10 Maarten Sierhuis (UvA) *Modeling and Simulating Work Practice*
11 Tom M. van Engers (VUA) *Knowledge Management*

2002

1 Nico Lassing (VUA) *Architecture-Level Modifiability Analysis*
2 Roelof van Zwol (UT) *Modelling and searching web-based document collections*
3 Henk Ernst Blok (UT) *Database Optimization Aspects for Information Retrieval*
4 Juan Roberto Castelo Valdueza (UU) *The Discrete Acyclic Digraph Markov Model in Data Mining*
5 Radu Serban (VUA) *The Private Cyberspace Modeling Electronic*
6 Laurens Mommers (UL) *Applied legal epistemology: Building a knowledge-based ontology of*
7 Peter Boncz (CWI) *Monet: A Next-Generation DBMS Kernel For Query-Intensive*
8 Jaap Gordijn (VUA) *Value Based Requirements Engineering: Exploring Innovative*
9 Willem-Jan van den Heuvel (KUB) *Integrating Modern Business Applications with Objectified Legacy*
10 Brian Sheppard (UM) *Towards Perfect Play of Scrabble*
11 Wouter C. A. Wijngaards (VUA) *Agent Based Modelling of Dynamics: Biological and Organisational Applications*

12 Albrecht Schmidt (UvA) *Processing XML in Database Systems*
13 Hongjing Wu (TUe) *A Reference Architecture for Adaptive Hypermedia Applications*
14 Wieke de Vries (UU) *Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems*
15 Rik Eshuis (UT) *Semantics and Verification of UML Activity Diagrams for Workflow Modelling*
16 Pieter van Langen (VUA) *The Anatomy of Design: Foundations, Models and Applications*
17 Stefan Manegold (UvA) *Understanding, Modeling, and Improving Main-Memory Database Performance*

2003

1 Heiner Stuckenschmidt (VUA) *Ontology-Based Information Sharing in Weakly Structured Environments*
2 Jan Broersen (VUA) *Modal Action Logics for Reasoning About Reactive Systems*
3 Martijn Schuemie (TUD) *Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy*
4 Milan Petkovic (UT) *Content-Based Video Retrieval Supported by Database Technology*
5 Jos Lehmann (UvA) *Causation in Artificial Intelligence and Law: A modelling approach*
6 Boris van Schooten (UT) *Development and specification of virtual environments*
7 Machiel Jansen (UvA) *Formal Explorations of Knowledge Intensive Tasks*
8 Yongping Ran (UM) *Repair Based Scheduling*
9 Rens Kortmann (UM) *The resolution of visually guided behaviour*
10 Andreas Lincke (UvT) *Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture*
11 Simon Keizer (UT) *Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks*
12 Roeland Ordelman (UT) *Dutch speech recognition in multimedia information retrieval*
13 Jeroen Donkers (UM) *Nosce Hostem: Searching with Opponent Models*
14 Stijn Hoppenbrouwers (KUN) *Freezing Language: Conceptualisation Processes across ICT-Supported Organisations*
15 Mathijs de Weerdt (TUD) *Plan Merging in Multi-Agent Systems*
16 Menzo Windhouwer (CWI) *Feature Grammar Systems: Incremental Maintenance of Indexes to Digital Media Warehouses*
17 David Jansen (UT) *Extensions of Statecharts with Probability, Time, and Stochastic Timing*
18 Levente Kocsis (UM) *Learning Search Decisions*

2004

1 Virginia Dignum (UU) *A Model for Organizational Interaction: Based on Agents, Founded in Logic*
2 Lai Xu (UvT) *Monitoring Multi-party Contracts for E-business*
3 Perry Groot (VUA) *A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving*
4 Chris van Aart (UvA) *Organizational Principles for Multi-Agent Architectures*
5 Viara Popova (EUR) *Knowledge discovery and monotonicity*
6 Bart-Jan Hommes (TUD) *The Evaluation of Business Process Modeling Techniques*
7 Elise Boltjes (UM) *Voorbeeldig onderwijs: voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes*
8 Joop Verbeek (UM) *Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiële gegevensuitwisseling en digitale expertise*
9 Martin Caminada (VUA) *For the Sake of the Argument: explorations into argument-based reasoning*
10 Suzanne Kabel (UvA) *Knowledge-rich indexing of learning-objects*
11 Michel Klein (VUA) *Change Management for Distributed Ontologies*
12 The Duy Bui (UT) *Creating emotions and facial expressions for embodied agents*
13 Wojciech Jamroga (UT) *Using Multiple Models of Reality: On Agents who Know how to Play*
14 Paul Harrenstein (UU) *Logic in Conflict. Logical Explorations in Strategic Equilibrium*
15 Arno Knobbe (UU) *Multi-Relational Data Mining*
16 Federico Divina (VUA) *Hybrid Genetic Relational Search for Inductive Learning*
17 Mark Winands (UM) *Informed Search in Complex Games*
18 Vania Bessa Machado (UvA) *Supporting the Construction of Qualitative Knowledge Models*
19 Thijs Westerveld (UT) *Using generative probabilistic models for multimedia retrieval*
20 Madelon Evers (Nyenrode) *Learning from Design: facilitating multidisciplinary design teams*

2005

1 Floor Verdenius (UvA) *Methodological Aspects of Designing Induction-Based Applications*
2 Erik van der Werf (UM) *AI techniques for the game of Go*
3 Franc Grootjen (RUN) *A Pragmatic Approach to the Conceptualisation of Language*
4 Nirvana Meratnia (UT) *Towards Database Support for Moving Object data*
5 Gabriel Infante-Lopez (UvA) *Two-Level Probabilistic Grammars for Natural Language Parsing*
6 Pieter Spronck (UM) *Adaptive Game AI*

7 Flavius Frasincar (TUe) *Hypermedia Presentation Generation for Semantic Web Information Systems*
8 Richard Vdovjak (TUe) *A Model-driven Approach for Building Distributed Ontology-based Web Applications*
9 Jeen Broekstra (VUA) *Storage, Querying and Inferencing for Semantic Web Languages*
10 Anders Bouwer (UvA) *Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments*
11 Elth Ogston (VUA) *Agent Based Matchmaking and Clustering: A Decentralized Approach to Search*
12 Csaba Boer (EUR) *Distributed Simulation in Industry*
13 Fred Hamburg (UL) *Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen*
14 Borys Omelayenko (VUA) *Web-Service configuration on the Semantic Web: Exploring how semantics meets pragmatics*
15 Tibor Bosse (VUA) *Analysis of the Dynamics of Cognitive Processes*
16 Joris Graaumans (UU) *Usability of XML Query Languages*
17 Boris Shishkov (TUD) *Software Specification Based on Re-usable Business Components*
18 Danielle Sent (UU) *Test-selection strategies for probabilistic networks*
19 Michel van Dartel (UM) *Situated Representation*
20 Cristina Coteanu (UL) *Cyber Consumer Law, State of the Art and Perspectives*
21 Wijnand Derks (UT) *Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics*

2006

1 Samuil Angelov (TUe) *Foundations of B2B Electronic Contracting*
2 Cristina Chisalita (VUA) *Contextual issues in the design and use of information technology in organizations*
3 Noor Christoph (UvA) *The role of metacognitive skills in learning to solve problems*
4 Marta Sabou (VUA) *Building Web Service Ontologies*
5 Cees Pierik (UU) *Validation Techniques for Object-Oriented Proof Outlines*
6 Ziv Baida (VUA) *Software-aided Service Bundling: Intelligent Methods & Tools for Graphical Service Modeling*
7 Marko Smiljanic (UT) *XML schema matching: balancing efficiency and effectiveness by means of clustering*
8 Eelco Herder (UT) *Forward, Back and Home Again: Analyzing User Behavior on the Web*
9 Mohamed Wahdan (UM) *Automatic Formulation of the Auditor's Opinion*
10 Ronny Siebes (VUA) *Semantic Routing in Peer-to-Peer Systems*
11 Joeri van Ruth (UT) *Flattening Queries over Nested Data Types*
12 Bert Bongers (VUA) *Interactivation: Towards an e-cology of people, our technological environment, and the arts*
13 Henk-Jan Lebbink (UU) *Dialogue and Decision Games for Information Exchanging Agents*
14 Johan Hoorn (VUA) *Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change*
15 Rainer Malik (UU) *CONAN: Text Mining in the Biomedical Domain*
16 Carsten Riggelsen (UU) *Approximation Methods for Efficient Learning of Bayesian Networks*
17 Stacey Nagata (UU) *User Assistance for Multitasking with Interruptions on a Mobile Device*
18 Valentin Zhizhkun (UvA) *Graph transformation for Natural Language Processing*
19 Birna van Riemsdijk (UU) *Cognitive Agent Programming: A Semantic Approach*
20 Marina Velikova (UvT) *Monotone models for prediction in data mining*
21 Bas van Gils (RUN) *Aptness on the Web*
22 Paul de Vrieze (RUN) *Fundaments of Adaptive Personalisation*
23 Ion Juvina (UU) *Development of Cognitive Model for Navigating on the Web*
24 Laura Hollink (VUA) *Semantic Annotation for Retrieval of Visual Resources*
25 Madalina Drugan (UU) *Conditional log-likelihood MDL and Evolutionary MCMC*
26 Vojkan Mihajlovic (UT) *Score Region Algebra: A Flexible Framework for Structured Information Retrieval*
27 Stefano Bocconi (CWI) *Vox Populi: generating video documentaries from semantically annotated media repositories*
28 Borkur Sigurbjornsson (UvA) *Focused Information Access using XML Element Retrieval*

2007

1 Kees Leune (UvT) *Access Control and Service-Oriented Architectures*
2 Wouter Teepe (RUG) *Reconciling Information Exchange and Confidentiality: A Formal Approach*
3 Peter Mika (VUA) *Social Networks and the Semantic Web*
4 Jurriaan van Diggelen (UU) *Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach*
5 Bart Schermer (UL) *Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance*
6 Gilad Mishne (UvA) *Applied Text Analytics for Blogs*
7 Natasa Jovanovic' (UT) *To Whom It May Concern: Addressee Identification in Face-to-Face Meetings*

8 Mark Hoogendoorn (VUA) *Modeling of Change in Multi-Agent Organizations*
9 David Mobach (VUA) *Agent-Based Mediated Service Negotiation*
10 Huib Aldewereld (UU) *Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols*
11 Natalia Stash (TUe) *Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System*
12 Marcel van Gerven (RUN) *Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty*
13 Rutger Rienks (UT) *Meetings in Smart Environments: Implications of Progressing Technology*
14 Niek Bergboer (UM) *Context-Based Image Analysis*
15 Joyca Lacroix (UM) *NIM: a Situated Computational Memory Model*
16 Davide Grossi (UU) *Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems*
17 Theodore Charitos (UU) *Reasoning with Dynamic Networks in Practice*
18 Bart Orriens (UvT) *On the development an management of adaptive business collaborations*
19 David Levy (UM) *Intimate relationships with artificial partners*
20 Slinger Jansen (UU) *Customer Configuration Updating in a Software Supply Network*
21 Karianne Vermaas (UU) *Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005*
22 Zlatko Zlatev (UT) *Goal-oriented design of value and process models from patterns*
23 Peter Barna (TUe) *Specification of Application Logic in Web Information Systems*
24 Georgina Ramírez Camps (CWI) *Structural Features in XML Retrieval*
25 Joost Schalken (VUA) *Empirical Investigations in Software Process Improvement*

2008

1 Katalin Boer-Sorbán (EUR) *Agent-Based Simulation of Financial Markets: A modular, continuous-time approach*
2 Alexei Sharpanskykh (VUA) *On Computer-Aided Methods for Modeling and Analysis of Organizations*
3 Vera Hollink (UvA) *Optimizing hierarchical menus: a usage-based approach*
4 Ander de Keijzer (UT) *Management of Uncertain Data: towards unattended integration*
5 Bela Mutschler (UT) *Modeling and simulating causal dependencies on process-aware information systems from a cost perspective*
6 Arjen Hommersom (RUN) *On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective*
7 Peter van Rosmalen (OU) *Supporting the tutor in the design and support of adaptive e-learning*
8 Janneke Bolt (UU) *Bayesian Networks: Aspects of Approximate Inference*
9 Christof van Nimwegen (UU) *The paradox of the guided user: assistance can be counter-effective*
10 Wauter Bosma (UT) *Discourse oriented summarization*
11 Vera Kartseva (VUA) *Designing Controls for Network Organizations: A Value-Based Approach*
12 Jozsef Farkas (RUN) *A Semiotically Oriented Cognitive Model of Knowledge Representation*
13 Caterina Carraciolo (UvA) *Topic Driven Access to Scientific Handbooks*
14 Arthur van Bunningen (UT) *Context-Aware Querying: Better Answers with Less Effort*
15 Martijn van Otterlo (UT) *The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains*
16 Henriette van Vugt (VUA) *Embodied agents from a user's perspective*
17 Martin Op 't Land (TUD) *Applying Architecture and Ontology to the Splitting and Allying of Enterprises*
18 Guido de Croon (UM) *Adaptive Active Vision*
19 Henning Rode (UT) *From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search*
20 Rex Arendsen (UvA) *Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven*
21 Krisztian Balog (UvA) *People Search in the Enterprise*
22 Henk Koning (UU) *Communication of IT-Architecture*
23 Stefan Visscher (UU) *Bayesian network models for the management of ventilator-associated pneumonia*
24 Zharko Aleksovski (VUA) *Using background knowledge in ontology matching*
25 Geert Jonker (UU) *Efficient and Equitable Exchange in Air Traffic Management Plan Repair using Spender-signed Currency*
26 Marijn Huijbregts (UT) *Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled*
27 Hubert Vogten (OU) *Design and Implementation Strategies for IMS Learning Design*
28 Ildiko Flesch (RUN) *On the Use of Independence Relations in Bayesian Networks*
29 Dennis Reidsma (UT) *Annotations and Subjective Machines: Of Annotators, Embodied Agents, Users, and Other Humans*

30 Wouter van Atteveldt (VUA) *Semantic Network Analysis: Techniques for Extracting, Representing and Querying Media Content*
31 Loes Braun (UM) *Pro-Active Medical Information Retrieval*
32 Trung H. Bui (UT) *Toward Affective Dialogue Management using Partially Observable Markov Decision Processes*
33 Frank Terpstra (UvA) *Scientific Workflow Design: theoretical and practical issues*
34 Jeroen de Knijf (UU) *Studies in Frequent Tree Mining*
35 Ben Torben Nielsen (UvT) *Dendritic morphologies: function shapes structure*

2009

1 Rasa Jurgelenaite (RUN) *Symmetric Causal Independence Models*
2 Willem Robert van Hage (VUA) *Evaluating Ontology-Alignment Techniques*
3 Hans Stol (UvT) *A Framework for Evidence-based Policy Making Using IT*
4 Josephine Nabukenya (RUN) *Improving the Quality of Organisational Policy Making using Collaboration Engineering*
5 Sietse Overbeek (RUN) *Bridging Supply and Demand for Knowledge Intensive Tasks: Based on Knowledge, Cognition, and Quality*
6 Muhammad Subianto (UU) *Understanding Classification*
7 Ronald Poppe (UT) *Discriminative Vision-Based Recovery and Recognition of Human Motion*
8 Volker Nannen (VUA) *Evolutionary Agent-Based Policy Analysis in Dynamic Environments*
9 Benjamin Kanagwa (RUN) *Design, Discovery and Construction of Service-oriented Systems*
10 Jan Wielemaker (UvA) *Logic programming for knowledge-intensive interactive applications*
11 Alexander Boer (UvA) *Legal Theory, Sources of Law & the Semantic Web*
12 Peter Massuthe (TUE, Humboldt-Universitaet zu Berlin) *Operating Guidelines for Services*
13 Steven de Jong (UM) *Fairness in Multi-Agent Systems*
14 Maksym Korotkiy (VUA) *From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)*
15 Rinke Hoekstra (UvA) *Ontology Representation: Design Patterns and Ontologies that Make Sense*
16 Fritz Reul (UvT) *New Architectures in Computer Chess*
17 Laurens van der Maaten (UvT) *Feature Extraction from Visual Data*
18 Fabian Groffen (CWI) *Armada, An Evolving Database System*
19 Valentin Robu (CWI) *Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets*
20 Bob van der Vecht (UU) *Adjustable Autonomy: Controling Influences on Decision Making*
21 Stijn Vanderlooy (UM) *Ranking and Reliable Classification*
22 Pavel Serdyukov (UT) *Search For Expertise: Going beyond direct evidence*
23 Peter Hofgesang (VUA) *Modelling Web Usage in a Changing Environment*
24 Annerieke Heuvelink (VUA) *Cognitive Models for Training Simulations*
25 Alex van Ballegooij (CWI) *RAM: Array Database Management through Relational Mapping*
26 Fernando Koch (UU) *An Agent-Based Model for the Development of Intelligent Mobile Services*
27 Christian Glahn (OU) *Contextual Support of social Engagement and Reflection on the Web*
28 Sander Evers (UT) *Sensor Data Management with Probabilistic Models*
29 Stanislav Pokraev (UT) *Model-Driven Semantic Integration of Service-Oriented Applications*
30 Marcin Zukowski (CWI) *Balancing vectorized query execution with bandwidth-optimized storage*
31 Sofiya Katrenko (UvA) *A Closer Look at Learning Relations from Text*
32 Rik Farenhorst (VUA) *Architectural Knowledge Management: Supporting Architects and Auditors*
33 Khiet Truong (UT) *How Does Real Affect Affect Affect Recognition In Speech?*
34 Inge van de Weerd (UU) *Advancing in Software Product Management: An Incremental Method Engineering Approach*
35 Wouter Koelewijn (UL) *Privacy en Politiegegevens: Over geautomatiseerde normatieve informatie-uitwisseling*
36 Marco Kalz (OUN) *Placement Support for Learners in Learning Networks*
37 Hendrik Drachsler (OUN) *Navigation Support for Learners in Informal Learning Networks*
38 Riina Vuorikari (OU) *Tags and self-organisation: a metadata ecology for learning resources in a multilingual context*
39 Christian Stahl (TUE, Humboldt-Universitaet zu Berlin) *Service Substitution: A Behavioral Approach Based on Petri Nets*
40 Stephan Raaijmakers (UvT) *Multinomial Language Learning: Investigations into the Geometry of Language*
41 Igor Berezhnyy (UvT) *Digital Analysis of Paintings*
42 Toine Bogers (UvT) *Recommender Systems for Social Bookmarking*
43 Virginia Nunes Leal Franqueira (UT) *Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients*

44 Roberto Santana Tapia (UT) *Assessing Business-IT Alignment in Networked Organizations*
45 Jilles Vreeken (UU) *Making Pattern Mining Useful*
46 Loredana Afanasiev (UvA) *Querying XML: Benchmarks and Recursion*

2010

1 Matthijs van Leeuwen (UU) *Patterns that Matter*
2 Ingo Wassink (UT) *Work flows in Life Science*
3 Joost Geurts (CWI) *A Document Engineering Model and Processing Framework for Multimedia documents*
4 Olga Kulyk (UT) *Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments*
5 Claudia Hauff (UT) *Predicting the Effectiveness of Queries and Retrieval Systems*
6 Sander Bakkes (UvT) *Rapid Adaptation of Video Game AI*
7 Wim Fikkert (UT) *Gesture interaction at a Distance*
8 Krzysztof Siewicz (UL) *Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments*
9 Hugo Kielman (UL) *A Politiele gegevensverwerking en Privacy, Naar een effectieve waarborging*
10 Rebecca Ong (UL) *Mobile Communication and Protection of Children*
11 Adriaan Ter Mors (TUD) *The world according to MARP: Multi-Agent Route Planning*
12 Susan van den Braak (UU) *Sensemaking software for crime analysis*
13 Gianluigi Folino (RUN) *High Performance Data Mining using Bio-inspired techniques*
14 Sander van Splunter (VUA) *Automated Web Service Reconfiguration*
15 Lianne Bodenstaff (UT) *Managing Dependency Relations in Inter-Organizational Models*
16 Sicco Verwer (TUD) *Efficient Identification of Timed Automata, theory and practice*
17 Spyros Kotoulas (VUA) *Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications*
18 Charlotte Gerritsen (VUA) *Caught in the Act: Investigating Crime by Agent-Based Simulation*
19 Henriette Cramer (UvA) *People's Responses to Autonomous and Adaptive Systems*
20 Ivo Swartjes (UT) *Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative*
21 Harold van Heerde (UT) *Privacy-aware data management by means of data degradation*
22 Michiel Hildebrand (CWI) *End-user Support for Access to Heterogeneous Linked Data*
23 Bas Steunebrink (UU) *The Logical Structure of Emotions*
24 Zulfiqar Ali Memon (VUA) *Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective*
25 Ying Zhang (CWI) *XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines*
26 Marten Voulon (UL) *Automatisch contracteren*
27 Arne Koopman (UU) *Characteristic Relational Patterns*
28 Stratos Idreos (CWI) *Database Cracking: Towards Auto-tuning Database Kernels*
29 Marieke van Erp (UvT) *Accessing Natural History: Discoveries in data cleaning, structuring, and retrieval*
30 Victor de Boer (UvA) *Ontology Enrichment from Heterogeneous Sources on the Web*
31 Marcel Hiel (UvT) *An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems*
32 Robin Aly (UT) *Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval*
33 Teduh Dirgahayu (UT) *Interaction Design in Service Compositions*
34 Dolf Trieschnigg (UT) *Proof of Concept: Concept-based Biomedical Information Retrieval*
35 Jose Janssen (OU) *Paving the Way for Lifelong Learning: Facilitating competence development through a learning path specification*
36 Niels Lohmann (TUe) *Correctness of services and their composition*
37 Dirk Fahland (TUe) *From Scenarios to components*
38 Ghazanfar Farooq Siddiqui (VUA) *Integrative modeling of emotions in virtual agents*
39 Mark van Assem (VUA) *Converting and Integrating Vocabularies for the Semantic Web*
40 Guillaume Chaslot (UM) *Monte-Carlo Tree Search*
41 Sybren de Kinderen (VUA) *Needs-driven service bundling in a multi-supplier setting: the computational e3-service approach*
42 Peter van Kranenburg (UU) *A Computational Approach to Content-Based Retrieval of Folk Song Melodies*
43 Pieter Bellekens (TUe) *An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain*
44 Vasilios Andrikopoulos (UvT) *A theory and model for the evolution of software services*
45 Vincent Pijpers (VUA) *e3alignment: Exploring Inter-Organizational Business-ICT Alignment*
46 Chen Li (UT) *Mining Process Model Variants: Challenges, Techniques, Examples*
47 Jahn-Takeshi Saito (UM) *Solving difficult game positions*

48 Bouke Huurnink (UvA) *Search in Audiovisual Broadcast Archives*

49 Alia Khairia Amin (CWI) *Understanding and supporting information seeking tasks in multiple sources*

50 Peter-Paul van Maanen (VUA) *Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention*

51 Edgar Meij (UvA) *Combining Concepts and Language Models for Information Access*

2011

1 Botond Cseke (RUN) *Variational Algorithms for Bayesian Inference in Latent Gaussian Models*

2 Nick Tinnemeier (UU) *Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language*

3 Jan Martijn van der Werf (TUe) *Compositional Design and Verification of Component-Based Information Systems*

4 Hado van Hasselt (UU) *Insights in Reinforcement Learning: Formal analysis and empirical evaluation of temporal-difference*

5 Base van der Raadt (VUA) *Enterprise Architecture Coming of Age: Increasing the Performance of an Emerging Discipline*

6 Yiwen Wang (TUe) *Semantically-Enhanced Recommendations in Cultural Heritage*

7 Yujia Cao (UT) *Multimodal Information Presentation for High Load Human Computer Interaction*

8 Nieske Vergunst (UU) *BDI-based Generation of Robust Task-Oriented Dialogues*

9 Tim de Jong (OU) *Contextualised Mobile Media for Learning*

10 Bart Bogaert (UvT) *Cloud Content Contention*

11 Dhaval Vyas (UT) *Designing for Awareness: An Experience-focused HCI Perspective*

12 Carmen Bratosin (TUe) *Grid Architecture for Distributed Process Mining*

13 Xiaoyu Mao (UvT) *Airport under Control. Multi-agent Scheduling for Airport Ground Handling*

14 Milan Lovric (EUR) *Behavioral Finance and Agent-Based Artificial Markets*

15 Marijn Koolen (UvA) *The Meaning of Structure: the Value of Link Evidence for Information Retrieval*

16 Maarten Schadd (UM) *Selective Search in Games of Different Complexity*

17 Jiyin He (UvA) *Exploring Topic Structure: Coherence, Diversity and Relatedness*

18 Mark Ponsen (UM) *Strategic Decision-Making in complex games*

19 Ellen Rusman (OU) *The Mind ' s Eye on Personal Profiles*

20 Qing Gu (VUA) *Guiding service-oriented software engineering: A view-based approach*

21 Linda Terlouw (TUD) *Modularization and Specification of Service-Oriented Systems*

22 Junte Zhang (UvA) *System Evaluation of Archival Description and Access*

23 Wouter Weerkamp (UvA) *Finding People and their Utterances in Social Media*

24 Herwin van Welbergen (UT) *Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior*

25 Syed Waqar ul Qounain Jaffry (VUA) *Analysis and Validation of Models for Trust Dynamics*

26 Matthijs Aart Pontier (VUA) *Virtual Agents for Human Communication: Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots*

27 Aniel Bhulai (VUA) *Dynamic website optimization through autonomous management of design patterns*

28 Rianne Kaptein (UvA) *Effective Focused Retrieval by Exploiting Query Context and Document Structure*

29 Faisal Kamiran (TUe) *Discrimination-aware Classification*

30 Egon van den Broek (UT) *Affective Signal Processing (ASP): Unraveling the mystery of emotions*

31 Ludo Waltman (EUR) *Computational and Game-Theoretic Approaches for Modeling Bounded Rationality*

32 Nees-Jan van Eck (EUR) *Methodological Advances in Bibliometric Mapping of Science*

33 Tom van der Weide (UU) *Arguing to Motivate Decisions*

34 Paolo Turrini (UU) *Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations*

35 Maaike Harbers (UU) *Explaining Agent Behavior in Virtual Training*

36 Erik van der Spek (UU) *Experiments in serious game design: a cognitive approach*

37 Adriana Burlutiu (RUN) *Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference*

38 Nyree Lemmens (UM) *Bee-inspired Distributed Optimization*

39 Joost Westra (UU) *Organizing Adaptation using Agents in Serious Games*

40 Viktor Clerc (VUA) *Architectural Knowledge Management in Global Software Development*

41 Luan Ibraimi (UT) *Cryptographically Enforced Distributed Data Access Control*

42 Michal Sindlar (UU) *Explaining Behavior through Mental State Attribution*

43 Henk van der Schuur (UU) *Process Improvement through Software Operation Knowledge*

44 Boris Reuderink (UT) *Robust Brain-Computer Interfaces*

45 Herman Stehouwer (UvT) *Statistical Language Models for Alternative Sequence Selection*

46 Beibei Hu (TUD) *Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work*
47 Azizi Bin Ab Aziz (VUA) *Exploring Computational Models for Intelligent Support of Persons with Depression*
48 Mark Ter Maat (UT) *Response Selection and Turn-taking for a Sensitive Artificial Listening Agent*
49 Andreea Niculescu (UT) *Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality*

2012

1 Terry Kakeeto (UvT) *Relationship Marketing for SMEs in Uganda*
2 Muhammad Umair (VUA) *Adaptivity, emotion, and Rationality in Human and Ambient Agent Models*
3 Adam Vanya (VUA) *Supporting Architecture Evolution by Mining Software Repositories*
4 Jurriaan Souer (UU) *Development of Content Management System-based Web Applications*
5 Marijn Plomp (UU) *Maturing Interorganisational Information Systems*
6 Wolfgang Reinhardt (OU) *Awareness Support for Knowledge Workers in Research Networks*
7 Rianne van Lambalgen (VUA) *When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions*
8 Gerben de Vries (UvA) *Kernel Methods for Vessel Trajectories*
9 Ricardo Neisse (UT) *Trust and Privacy Management Support for Context-Aware Service Platforms*
10 David Smits (TUe) *Towards a Generic Distributed Adaptive Hypermedia Environment*
11 J. C. B. Rantham Prabhakara (TUe) *Process Mining in the Large: Preprocessing, Discovery, and Diagnostics*
12 Kees van der Sluijs (TUe) *Model Driven Design and Data Integration in Semantic Web Information Systems*
13 Suleman Shahid (UvT) *Fun and Face: Exploring non-verbal expressions of emotion during playful interactions*
14 Evgeny Knutov (TUe) *Generic Adaptation Framework for Unifying Adaptive Web-based Systems*
15 Natalie van der Wal (VUA) *Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes*
16 Fiemke Both (VUA) *Helping people by understanding them: Ambient Agents supporting task execution and depression treatment*
17 Amal Elgammal (UvT) *Towards a Comprehensive Framework for Business Process Compliance*
18 Eltjo Poort (VUA) *Improving Solution Architecting Practices*
19 Helen Schonenberg (TUe) *What's Next? Operational Support for Business Process Execution*
20 Ali Bahramisharif (RUN) *Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing*
21 Roberto Cornacchia (TUD) *Querying Sparse Matrices for Information Retrieval*
22 Thijs Vis (UvT) *Intelligence, politie en veiligheidsdienst: verenigbare grootheden?*
23 Christian Muehl (UT) *Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction*
24 Laurens van der Werff (UT) *Evaluation of Noisy Transcripts for Spoken Document Retrieval*
25 Silja Eckartz (UT) *Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application*
26 Emile de Maat (UvA) *Making Sense of Legal Text*
27 Hayrettin Gurkok (UT) *Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games*
28 Nancy Pascall (UvT) *Engendering Technology Empowering Women*
29 Almer Tigelaar (UT) *Peer-to-Peer Information Retrieval*
30 Alina Pommeranz (TUD) *Designing Human-Centered Systems for Reflective Decision Making*
31 Emily Bagarukayo (RUN) *A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure*
32 Wietske Visser (TUD) *Qualitative multi-criteria preference representation and reasoning*
33 Rory Sie (OUN) *Coalitions in Cooperation Networks (COCOON)*
34 Pavol Jancura (RUN) *Evolutionary analysis in PPI networks and applications*
35 Evert Haasdijk (VUA) *Never Too Old To Learn: On-line Evolution of Controllers in Swarm- and Modular Robotics*
36 Denis Ssebugwawo (RUN) *Analysis and Evaluation of Collaborative Modeling Processes*
37 Agnes Nakakawa (RUN) *A Collaboration Process for Enterprise Architecture Creation*
38 Selmar Smit (VUA) *Parameter Tuning and Scientific Testing in Evolutionary Algorithms*
39 Hassan Fatemi (UT) *Risk-aware design of value and coordination networks*
40 Agus Gunawan (UvT) *Information Access for SMEs in Indonesia*
41 Sebastian Kelle (OU) *Game Design Patterns for Learning*
42 Dominique Verpoorten (OU) *Reflection Amplifiers in self-regulated Learning*
43 Anna Tordai (VUA) *On Combining Alignment Techniques*

44 Benedikt Kratz (UvT) *A Model and Language for Business-aware Transactions*
45 Simon Carter (UvA) *Exploration and Exploitation of Multilingual Data for Statistical Machine Translation*
46 Manos Tsagkias (UvA) *Mining Social Media: Tracking Content and Predicting Behavior*
47 Jorn Bakker (TUe) *Handling Abrupt Changes in Evolving Time-series Data*
48 Michael Kaisers (UM) *Learning against Learning: Evolutionary dynamics of reinforcement learning algorithms in strategic interactions*
49 Steven van Kervel (TUD) *Ontologogy driven Enterprise Information Systems Engineering*
50 Jeroen de Jong (TUD) *Heuristics in Dynamic Sceduling: a practical framework with a case study in elevator dispatching*

2013

1 Viorel Milea (EUR) *News Analytics for Financial Decision Support*
2 Erietta Liarou (CWI) *MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing*
3 Szymon Klarman (VUA) *Reasoning with Contexts in Description Logics*
4 Chetan Yadati (TUD) *Coordinating autonomous planning and scheduling*
5 Dulce Pumareja (UT) *Groupware Requirements Evolutions Patterns*
6 Romulo Goncalves (CWI) *The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience*
7 Giel van Lankveld (UvT) *Quantifying Individual Player Differences*
8 Robbert-Jan Merk (VUA) *Making enemies: cognitive modeling for opponent agents in fighter pilot simulators*
9 Fabio Gori (RUN) *Metagenomic Data Analysis: Computational Methods and Applications*
10 Jeewanie Jayasinghe Arachchige (UvT) *A Unified Modeling Framework for Service Design*
11 Evangelos Pournaras (TUD) *Multi-level Reconfigurable Self-organization in Overlay Services*
12 Marian Razavian (VUA) *Knowledge-driven Migration to Services*
13 Mohammad Safiri (UT) *Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly*
14 Jafar Tanha (UvA) *Ensemble Approaches to Semi-Supervised Learning Learning*
15 Daniel Hennes (UM) *Multiagent Learning: Dynamic Games and Applications*
16 Eric Kok (UU) *Exploring the practical benefits of argumentation in multi-agent deliberation*
17 Koen Kok (VUA) *The PowerMatcher: Smart Coordination for the Smart Electricity Grid*
18 Jeroen Janssens (UvT) *Outlier Selection and One-Class Classification*
19 Renze Steenhuizen (TUD) *Coordinated Multi-Agent Planning and Scheduling*
20 Katja Hofmann (UvA) *Fast and Reliable Online Learning to Rank for Information Retrieval*
21 Sander Wubben (UvT) *Text-to-text generation by monolingual machine translation*
22 Tom Claassen (RUN) *Causal Discovery and Logic*
23 Patricio de Alencar Silva (UvT) *Value Activity Monitoring*
24 Haitham Bou Ammar (UM) *Automated Transfer in Reinforcement Learning*
25 Agnieszka Anna Latoszek-Berendsen (UM) *Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System*
26 Alireza Zarghami (UT) *Architectural Support for Dynamic Homecare Service Provisioning*
27 Mohammad Huq (UT) *Inference-based Framework Managing Data Provenance*
28 Frans van der Sluis (UT) *When Complexity becomes Interesting: An Inquiry into the Information eXperience*
29 Iwan de Kok (UT) *Listening Heads*
30 Joyce Nakatumba (TUe) *Resource-Aware Business Process Management: Analysis and Support*
31 Dinh Khoa Nguyen (UvT) *Blueprint Model and Language for Engineering Cloud Applications*
32 Kamakshi Rajagopal (OUN) *Networking For Learning: The role of Networking in a Lifelong Learner's Professional Development*
33 Qi Gao (TUD) *User Modeling and Personalization in the Microblogging Sphere*
34 Kien Tjin-Kam-Jet (UT) *Distributed Deep Web Search*
35 Abdallah El Ali (UvA) *Minimal Mobile Human Computer Interaction*
36 Than Lam Hoang (TUe) *Pattern Mining in Data Streams*
37 Dirk Börner (OUN) *Ambient Learning Displays*
38 Eelco den Heijer (VUA) *Autonomous Evolutionary Art*
39 Joop de Jong (TUD) *A Method for Enterprise Ontology based Design of Enterprise Information Systems*
40 Pim Nijssen (UM) *Monte-Carlo Tree Search for Multi-Player Games*
41 Jochem Liem (UvA) *Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning*
42 Léon Planken (TUD) *Algorithms for Simple Temporal Reasoning*
43 Marc Bron (UvA) *Exploration and Contextualization through Interaction and Concepts*

2014

1 Nicola Barile (UU) *Studies in Learning Monotone Models from Data*

2 Fiona Tuliyano (RUN) *Combining System Dynamics with a Domain Modeling Method*
3 Sergio Raul Duarte Torres (UT) *Information Retrieval for Children: Search Behavior and Solutions*
4 Hanna Jochmann-Mannak (UT) *Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation*
5 Jurriaan van Reijsen (UU) *Knowledge Perspectives on Advancing Dynamic Capability*
6 Damian Tamburri (VUA) *Supporting Networked Software Development*
7 Arya Adriansyah (TUe) *Aligning Observed and Modeled Behavior*
8 Samur Araujo (TUD) *Data Integration over Distributed and Heterogeneous Data Endpoints*
9 Philip Jackson (UvT) *Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language*
10 Ivan Salvador Razo Zapata (VUA) *Service Value Networks*
11 Janneke van der Zwaan (TUD) *An Empathic Virtual Buddy for Social Support*
12 Willem van Willigen (VUA) *Look Ma, No Hands: Aspects of Autonomous Vehicle Control*
13 Arlette van Wissen (VUA) *Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains*
14 Yangyang Shi (TUD) *Language Models With Meta-information*
15 Natalya Mogles (VUA) *Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare*
16 Krystyna Milian (VUA) *Supporting trial recruitment and design by automatically interpreting eligibility criteria*
17 Kathrin Dentler (VUA) *Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability*
18 Mattijs Ghijsen (UvA) *Methods and Models for the Design and Study of Dynamic Agent Organizations*
19 Vinicius Ramos (TUe) *Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support*
20 Mena Habib (UT) *Named Entity Extraction and Disambiguation for Informal Text: The Missing Link*
21 Kassidy Clark (TUD) *Negotiation and Monitoring in Open Environments*
22 Marieke Peeters (UU) *Personalized Educational Games: Developing agent-supported scenario-based training*
23 Eleftherios Sidirourgos (UvA/CWI) *Space Efficient Indexes for the Big Data Era*
24 Davide Ceolin (VUA) *Trusting Semi-structured Web Data*
25 Martijn Lappenschaar (RUN) *New network models for the analysis of disease interaction*
26 Tim Baarslag (TUD) *What to Bid and When to Stop*
27 Rui Jorge Almeida (EUR) *Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty*
28 Anna Chmielowiec (VUA) *Decentralized k-Clique Matching*
29 Jaap Kabbedijk (UU) *Variability in Multi-Tenant Enterprise Software*
30 Peter de Cock (UvT) *Anticipating Criminal Behaviour*
31 Leo van Moergestel (UU) *Agent Technology in Agile Multiparallel Manufacturing and Product Support*
32 Naser Ayat (UvA) *On Entity Resolution in Probabilistic Data*
33 Tesfa Tegegne (RUN) *Service Discovery in eHealth*
34 Christina Manteli (VUA) *The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems*
35 Joost van Ooijen (UU) *Cognitive Agents in Virtual Worlds: A Middleware Design Approach*
36 Joos Buijs (TUe) *Flexible Evolutionary Algorithms for Mining Structured Process Models*
37 Maral Dadvar (UT) *Experts and Machines United Against Cyberbullying*
38 Danny Plass-Oude Bos (UT) *Making brain-computer interfaces better: improving usability through post-processing*
39 Jasmina Maric (UvT) *Web Communities, Immigration, and Social Capital*
40 Walter Omona (RUN) *A Framework for Knowledge Management Using ICT in Higher Education*
41 Frederic Hogenboom (EUR) *Automated Detection of Financial Events in News Text*
42 Carsten Eijckhof (CWI/TUD) *Contextual Multidimensional Relevance Models*
43 Kevin Vlaanderen (UU) *Supporting Process Improvement using Method Increments*
44 Paulien Meesters (UvT) *Intelligent Blauw: Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden*
45 Birgit Schmitz (OUN) *Mobile Games for Learning: A Pattern-Based Approach*
46 Ke Tao (TUD) *Social Web Data Analytics: Relevance, Redundancy, Diversity*
47 Shangsong Liang (UvA) *Fusion and Diversification in Information Retrieval*

2015

1 Niels Netten (UvA) *Machine Learning for Relevance of Information in Crisis Response*
2 Faiza Bukhsh (UvT) *Smart auditing: Innovative Compliance Checking in Customs Controls*

3 Twan van Laarhoven (RUN) *Machine learning for network data*
4 Howard Spoelstra (OUN) *Collaborations in Open Learning Environments*
5 Christoph Bösch (UT) *Cryptographically Enforced Search Pattern Hiding*
6 Farideh Heidari (TUD) *Business Process Quality Computation: Computing Non-Functional Requirements to Improve Business Processes*
7 Maria-Hendrike Peetz (UvA) *Time-Aware Online Reputation Analysis*
8 Jie Jiang (TUD) *Organizational Compliance: An agent-based model for designing and evaluating organizational interactions*
9 Randy Klaassen (UT) *HCI Perspectives on Behavior Change Support Systems*
10 Henry Hermans (OUN) *OpenU: design of an integrated system to support lifelong learning*
11 Yongming Luo (TUe) *Designing algorithms for big graph datasets: A study of computing bisimulation and joins*
12 Julie M. Birkholz (VUA) *Modi Operandi of Social Network Dynamics: The Effect of Context on Scientific Collaboration Networks*
13 Giuseppe Procaccianti (VUA) *Energy-Efficient Software*
14 Bart van Straalen (UT) *A cognitive approach to modeling bad news conversations*
15 Klaas Andries de Graaf (VUA) *Ontology-based Software Architecture Documentation*
16 Changyun Wei (UT) *Cognitive Coordination for Cooperative Multi-Robot Teamwork*
17 André van Cleeff (UT) *Physical and Digital Security Mechanisms: Properties, Combinations and Trade-offs*
18 Holger Pirk (CWI) *Waste Not, Want Not!: Managing Relational Data in Asymmetric Memories*
19 Bernardo Tabuenca (OUN) *Ubiquitous Technology for Lifelong Learners*
20 Loïs Vanhée (UU) *Using Culture and Values to Support Flexible Coordination*
21 Sibren Fetter (OUN) *Using Peer-Support to Expand and Stabilize Online Learning*
22 Zhemin Zhu (UT) *Co-occurrence Rate Networks*
23 Luit Gazendam (VUA) *Cataloguer Support in Cultural Heritage*
24 Richard Berendsen (UvA) *Finding People, Papers, and Posts: Vertical Search Algorithms and Evaluation*
25 Steven Woudenberg (UU) *Bayesian Tools for Early Disease Detection*
26 Alexander Hogenboom (EUR) *Sentiment Analysis of Text Guided by Semantics and Structure*
27 Sándor Héman (CWI) *Updating compressed column-stores*
28 Janet Bagorogoza (TiU) *Knowledge Management and High Performance: The Uganda Financial Institutions Model for HPO*
29 Hendrik Baier (UM) *Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains*
30 Kiavash Bahreini (OUN) *Real-time Multimodal Emotion Recognition in E-Learning*
31 Yakup Koç (TUD) *On Robustness of Power Grids*
32 Jerome Gard (UL) *Corporate Venture Management in SMEs*
33 Frederik Schadd (UM) *Ontology Mapping with Auxiliary Resources*
34 Victor de Graaff (UT) *Geosocial Recommender Systems*
35 Junchao Xu (TUD) *Affective Body Language of Humanoid Robots: Perception and Effects in Human Robot Interaction*

2016

1 Syed Saiden Abbas (RUN) *Recognition of Shapes by Humans and Machines*
2 Michiel Christiaan Meulendijk (UU) *Optimizing medication reviews through decision support: prescribing a better pill to swallow*
3 Maya Sappelli (RUN) *Knowledge Work in Context: User Centered Knowledge Worker Support*
4 Laurens Rietveld (VUA) *Publishing and Consuming Linked Data*
5 Evgeny Sherkhonov (UvA) *Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers*
6 Michel Wilson (TUD) *Robust scheduling in an uncertain environment*
7 Jeroen de Man (VUA) *Measuring and modeling negative emotions for virtual training*
8 Matje van de Camp (TiU) *A Link to the Past: Constructing Historical Social Networks from Unstructured Data*
9 Archana Nottamkandath (VUA) *Trusting Crowdsourced Information on Cultural Artefacts*
10 George Karafotias (VUA) *Parameter Control for Evolutionary Algorithms*
11 Anne Schuth (UvA) *Search Engines that Learn from Their Users*
12 Max Knobbout (UU) *Logics for Modelling and Verifying Normative Multi-Agent Systems*
13 Nana Baah Gyan (VU) *The Web, Speech Technologies and Rural Development in West Africa: An ICT4D Approach*
14 Ravi Khadka (UU) *Revisiting Legacy Software System Modernization*
15 Steffen Michels (RUN) *Hybrid Probabilistic Logics: Theoretical Aspects, Algorithms and Experiments*
16 Guangliang Li (UvA) *Socially Intelligent Autonomous Agents that Learn from Human Reward*
17 Berend Weel (VUA) *Towards Embodied Evolution of Robot Organisms*
18 Albert Meroño Peñuela (VUA) *Refining Statistical Data on the Web*

19 Julia Efremova (TUe) *Mining Social Structures from Genealogical Data*
20 Daan Odijk (UvA) *Context & Semantics in News & Web Search*
21 Alejandro Moreno Célleri (UT) *From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground*
22 Grace Lewis (VU) *Software Architecture Strategies for Cyber-Foraging Systems*
23 Fei Cai (UvA) *Query Auto Completion in Information Retrieval*
24 Brend Wanders (UT) *Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach*
25 Julia Kiseleva (TUe) *Using Contextual Information to Understand Searching and Browsing Behavior*
26 Dilhan Thilakarathne (VU) *In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains*
27 Wen Li (TUD) *Understanding Geo-spatial Information on Social Media*
28 Mingxin Zhang (TUD) *Large-scale Agent-based Social Simulation - A study on epidemic prediction and control*
29 Nicolas Höning (CWI/TUD) *Peak reduction in decentralised electricity systems - Markets and prices for flexible planning*
30 Ruud Mattheij (UvT) *The Eyes Have It*
31 Mohammadreza Khelghati (UT) *Deep web content monitoring*
32 Eelco Vriezekolk (UT) *Assessing Telecommunication Service Availability Risks for Crisis Organisations*
33 Peter Bloem (UvA) *Single Sample Statistics, exercises in learning from just one example*
34 Dennis Schunselaar (TUe) *Configurable Process Trees: Elicitation, Analysis, and Enactment*
35 Zhaochun Ren (UvA) *Monitoring Social Media: Summarization, Classification and Recommendation*
36 Daphne Karreman (UT) *Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies*
37 Giovanni Sileno (UvA) *Aligning Law and Action: a conceptual and computational inquiry*
38 Andrea Minuto (UT) *MATERIALS THAT MATTER: Smart Materials meet Art & Interaction Design*
39 Merijn Bruijnes (UT) *Believable Suspect Agents: Response and Interpersonal Style Selection for an Artificial Suspect*
40 Christian Detweiler (TUD) *Accounting for Values in Design*
41 Thomas King (TUD) *Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance*
42 Spyros Martzoukos (UvA) *Combinatorial and Compositional Aspects of Bilingual Aligned Corpora*
43 Saskia Koldijk (RUN) *Context-Aware Support for Stress Self-Management: From Theory to Practice*
44 Thibault Sellam (UvA) *Automatic Assistants for Database Exploration*
45 Bram van de Laar (UT) *Experiencing Brain-Computer Interface Control*
46 Jorge Gallego Perez (UT) *Robots to Make you Happy*
47 Christina Weber (UL) *Real-time foresight: Preparedness for dynamic innovation networks*
48 Tanja Buttler (TUD) *Collecting Lessons Learned*
49 Gleb Polevoy (TUD) *Participation and Interaction in Projects. A Game-Theoretic Analysis*
50 Yan Wang (UVT) *The Bridge of Dreams: Towards a Method for Operational Performance Alignment in IT-enabled Service Supply Chains*

2017

1 Jan-Jaap Oerlemans (UL) *Investigating Cybercrime*
2 Sjoerd Timmer (UU) *Designing and Understanding Forensic Bayesian Networks using Argumentation*
3 Daniël Harold Telgen (UU) *Grid Manufacturing: A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines*
4 Mrunal Gawade (CWI) *Multi-core Parallelism in a Column-store*
5 Mahdieh Shadi (UvA) *Collaboration Behavior*
6 Damir Vandic (EUR) *Intelligent Information Systems for Web Product Search*
7 Roel Bertens (UU) *Insight in Information: from Abstract to Anomaly*
8 Rob Konijn (VUA) *Detecting Interesting Differences: Data Mining in Health Insurance Data using Outlier Detection and Subgroup Discovery*
9 Dong Nguyen (UT) *Text as Social and Cultural Data: A Computational Perspective on Variation in Text*
10 Robby van Delden (UT) *(Steering) Interactive Play Behavior*
11 Florian Kunneman (RUN) *Modelling patterns of time and emotion in Twitter #anticipointment*
12 Sander Leemans (UT) *Robust Process Mining with Guarantees*
13 Gijs Huisman (UT) *Social Touch Technology: Extending the reach of social touch through haptic technology*
14 Shoshannah Tekofsky (UvT) *You Are Who You Play You Are: Modelling Player Traits from Video Game Behavior*
15 Peter Berck (RUN) *Memory-Based Text Correction*
16 Aleksandr Chuklin (UvA) *Understanding and Modeling Users of Modern Search Engines*

As search is being used by billions of people, modern search engines are becoming more and more complex. And complexity does not just come from the algorithms. Richer and richer content is being added to search engine result pages: news and sports results, definitions and translations, images and videos. Many such elements are added by search engines in their attempt to stand out from the competition by providing a superior user experience. However, the more complex search engines become, the harder it gets to understand users and their interactions with result pages, and to measure the quality of the user experience. In this thesis we address exactly this topic.