SUPPORTING THE DISCOVERY OF LONG-TAIL RESOURCES ON THE WEB

JÖRG SCHLÖTTERER

Doctoral Thesis
March 2020

Chair of Data Science
Faculty of Computer Science and Mathematics
University of Passau

UNIVERSITÄT PASSAU
Fakultät für Informatik und Mathematik

ABSTRACT

A plethora of resources made available via retrieval systems in digital libraries remains untapped in the so called long tail of the Web. These long-tail websites get considerably less visits than major Web hubs. Zero-effort queries ease the discovery of long-tail resources by proactively retrieving and presenting information based on a user's context. However, zero-effort queries over existing digital library structures are challenging, since the underlying retrieval system is only accessible via an API. The information need must be expressed by a query, instead of optimizing the ranking between context and resources in the retrieval system directly. We address three research questions that arise from replacing the user information seeking process by zero-effort queries.

Our first question addresses the transformation of a user query to an automatic query, derived from the context. We present means to 1) identify the relevant context on different levels of granularity, 2) derive an information need from the context via keyword extraction and personalization and 3) express this information need in a query scheme that avoids over- or under-specified queries. We address the cold start problem with an approach to bootstrap user profiles from social media, even for passive users.

With the second question, we address the presentation of resources in zero-effort query scenarios, presenting guidelines for presentation interfaces in the browser and a visualization of the triadic relationship between context, query and results. QueryCrumbs, a compact query history visualization supports recalling information found in the past and exploratory search by visualizing qualitative and quantitative query similarity.

Our last question addresses the gap between (simple) keyword queries and the representation of resources by rich and complex metadata. We investigate and extend feature representation learning techniques centered around the skip-gram model with negative sampling. Finally, we present an approach to learn representations from network and text jointly that can cope with the partial absence of one modality.

Experimental results show close to human performance of our zero-effort query and user profile generation approach and visualizations to be helpful in terms of transparency, efficiency and support for exploratory search. These results indicate that the proposed zero-effort query approach indeed eases the discovery of long-tail resources and the accompanying visualizations further facilitate this process. The joint representation model provides a first step to bridge the gap between query and resource representation and we plan to follow and investigate this route further in the future.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Part I

TRANSFORMING THE INFORMATION SEEKING PROCESS ON THE WEB TO ZERO-EFFORT QUERIES

1 INTRODUCTION

Scientific, cultural and educational resources often remain untapped in the so called long tail of the Web, i.e., sites which get considerably less visits than the major Web hubs like Google or Facebook. This long-tail content is niche content, which is particularly relevant to a small peer group, but still of potential interest to a broader audience. Although the content is of potential interest for a broader audience, users outside the small peer group may not be aware of its existence, nor do they know the access points to those isolated silos. The long-tail content we address in this thesis comprises resources of scientific, cultural and educational providers, such as scientific libraries, museum archives or art collections. We collectively refer to those as digital libraries.

Besides being hidden in the long tail of the Web, a further issue with digital libraries is that search in those archives is fundamentally different from regular Web-search. While Web-search typically comprises fact-finding like retrieval performed on full-text documents, search behavior in digital libraries is exploratory with keyword-based retrieval over object metadata. Traditionally, the core access point to a library has been the library catalog. In recent years, objects in these library catalogs have been extended with more and more metadata, in order to foster sophisticated search techniques, including relevance ranking or faceted search. While those sophisticated search techniques (and corresponding interfaces) provide great opportunities for specialists, they can be a burden for less experienced users [40].

Zero-effort queries, also known as just-in-time retrieval, are a viable way to ease the discovery of long-tail resources. They have been shown to increase the amount of information viewed by proactively retrieving information, based on a user's context and presenting results in an unobtrusive manner. Typically, zero-effort query systems take the current context as input and aim to present relevant resources, i.e. they aim to find a ranking between context and resources. We aim to reuse existing retrieval systems, which requires transforming the context into an appropriate representation to be used as input by the retrieval system, instead of optimizing the ranking between context and resources directly.

For this transformation, we adapt Broder's model of the information seeking process [26] as depicted in figure 1.1 on the following page. In Broder's original model, an information need arises within a certain task. This information need is then verbalized (mentally) and finally formulated as a query to the search engine. We need to automate the user's steps until a query is formulated, based on the user's context.

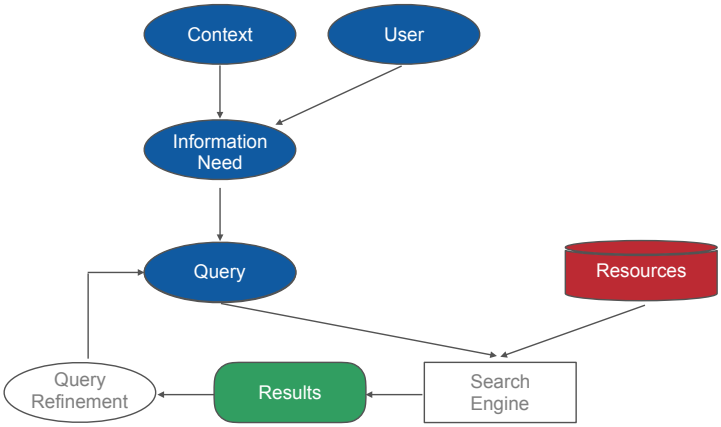


Figure 1.1: Adaption of Broder's information seeking process model to zero-effort queries. Colored parts are adapted/addressed by research questions.

This automation is colored blue in the figure and replaces the original user steps. The remaining parts in the figure stay the same as with the original model. Resource representation and result presentation are also addressed in this thesis and highlighted by color.

1.1 RESEARCH QUESTIONS

This section presents the research questions of this thesis. They are colored according to part(s) covered in figure 1.1. The available information sources to the just mentioned transformation of the user's steps to formulate a query to zero-effort queries are the current context (i.e., a Web page) and past user interactions (user profile). Therefore, we formulate our first research questions as follows:

RQ 1

How to transform the user's context into a query that retrieves relevant results?

This transformation requires to 1) identify the relevant context, 2) formulate an information need (based on context and user profile) and 3) represent this information need as a query to the search engine. While reusing existing retrieval systems, we aim to be search engine agnostic. That is, we do not optimize for a particular search engine but aim for a generic approach. In turn, we cannot rely on specific features or available metadata attributes of a particular search engine. RQ 1 is addressed in part ii on page 13.

When the user is actively seeking information, she expects to be presented with results. In zero-effort queries, the search process is fully automatic and the user may not want an interruption by result presentation, which leads to the second research question:

RQ 2

How to adapt result presentation to zero-effort queries?

This question comprises how to notify the user about retrieved results (and present them, if the user is interested). Further, we investigate how to increase transparency of the search process. We therefore evaluate visual explanations of the relationship between context, query and results. Further, we address the problem of remembering queries, which is even more prevalent in zero-effort queries (as the query is not formulated by the user, but automatically) by a query history visualization. This query history visualization is further intended to increase transparency of the search process for search experts. RQ 2 is addressed in part iii on page 87.

Since we aim for a search engine agnostic approach, our queries are limited to simple (boolean) keyword queries, without making use of particular search engine features. That is, we cannot limit queries to (provider-)specific metadata fields such as "keywords" or "subject", but query processing needs to be handled by the search engine provider. However, in particular digital libraries in the long tail of the Web, such as small museums or archives cannot afford complex query rewriting or metadata field mapping methods. We therefore seek a solution to a seamless integration of (simple) keyword based queries and complex metadata-based retrieval. As a first step towards this goal, we aim for a unified representation of resources, regardless the information source (document- or metadata-based). Accordingly, we formulate the third research question as:

RQ 3

How to represent resources to facilitate keyword-based search?

This question is not only relevant to zero-effort queries, but also to non-expert users. While these users are familiar with simple, "Google-like" search interfaces, they may not be able to exploit the rich and complex options of a sophisticated search interface to a specialized collection. These users would equally benefit from a seamless integration of simple keyword queries and complex metadata-based retrieval. RQ 3 is addressed in part iv on page 143.

1.2 CONTRIBUTIONS

The contributions of this thesis are the following (grouped by research questions):

RQ 1 - How to transform the user's context into a query that retrieves relevant results?

- **Transforming the process of user query formulation to zero-effort queries:** We present an approach to fully automatize the process of user query formulation from identifying an information need to finally formulating this need as query to a search system. In this line, we provide means to identify the relevant context within a Web page, how to use this context to formulate an information need and how to express this information need as query to the search system. Experimental results show that our approach allows to formulate zero-effort queries that perform better in terms of precision than user formulated queries.

- **Consumer profile generation from online social networks:** We present an approach to construct user profiles from online social networks, even for passive users. Passive users do not actively post content themselves, but express their interest, e.g., by following other users. With Twitter as a showcase, we demonstrate that followee lists (i.e., other users the user of interest follows) provide a sufficient basis for profile construction. Experimental results verify the high quality of the resulting profiles.

RQ 2 - How to adapt result presentation to zero-effort queries?

- **Visual explanation of zero-effort query results:** We visualize the triadic relationship between context, query and retrieved results. In experimental results, these visual explanations showed beneficial for efficiency: users were able to judge results faster.

- **QueryCrumbs query history visualization:** QueryCrumbs are a simple-to-understand visualization for accessing, altering and re-submitting previously issued queries. Their usability without instructions was confirmed in a formative user study with novices. A long-term study indicated actual uptake of the visualization and an evaluation with experts revealed that they can gain insights into search engine behavior, thus increasing transparency.

RQ 3 - How to represent resources to facilitate keyword-based search?

- **Evaluation of random-walk based network embedding modifications:** A thorough evaluation of modifications to random walk based network embedding revealed that they provide little to no gains in terms of node classification performance. The evaluated modifications comprise changes in the random walk strategy and abstractions of the graph on different levels. For both types, we compared methods available from the literature and complemented them with our own.

- **Joint embedding of network and text:** We present a joint model for embedding nodes in a network, based on both, their connections in the network and their textual content. While this model cannot outperform a naive concatenation of separately trained embeddings, it can still be trained if (partial) information is missing (e.g., no text is available for some nodes).

1.3 PUBLICATIONS

This section summarizes my own and joint publications and how they relate to this thesis.

The description of a first prototype to demonstrate the feasibility of zero-effort queries for long-tail content on the Web was presented in:

> Jörg Schlötterer, Christin Seifert, and Michael Granitzer. "Web-based just-in-time retrieval for cultural content." In: *PATCH14: Proceedings of the 7th International ACM Workshop on Personalized Access to Cultural Heritage*. 2014 (cf. [165])

The initial concept of different granularity levels within a Web page and how they can be made use of for query generation was presented in the following publication. Section 2.2 on page 15 presents the refined and final concept of contextual granularity levels and how they are used in this thesis.

> Jörg Schlötterer. "From Context to Query." In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. 2015 (cf. [164], 1st place student research competition)

The following publication presents the collection of a dataset for zero-effort queries and a showcase how to use that dataset for query generation. The contents of this publication are described in chapter 3 on page 21. While the focus of the publication is on the dataset collection, the focus of chapter 3 on page 21 is on the dataset analysis and query generation.

> Christin Seifert, Jörg Schlötterer, and Michael Granitzer. "Towards a Feature-rich Data Set for Personalized Access to Long-tail Content." In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. 2015 (cf. [175])

The following publications contain summaries of the query construction process on paragraph level (section 4.5 on page 52) and its performance. The second publication further contains brief results of focus paragraph extraction and detection (section 4.3 on page 39 and section 4.4 on page 46) and query personalization (section 4.6 on page 62).

> Jörg Schlötterer, Christin Seifert, and Michael Granitzer. "Supporting Web Surfers in Finding Related Material in Digital Library Repositories." In: *Research and Advanced Technology for Digital Libraries*. 2016 (cf. [166])

> Christin Seifert, Werner Bailer, Thomas Orgel, Louis Gantner, Roman Kern, Hermann Ziak, Albin Petit, Jörg Schlötterer, Stefan Zwicklbauer, and Michael Granitzer. "Ubiquitous Access to Digital Cultural Heritage." In: *J. Comput. Cult. Herit.* 10.1 (2017). ISSN: 1556-4673 (cf. [173])

Paragraph extraction (section 4.3 on page 39) has further been briefly described in the following publication. This publication also contains a detailed description of focus paragraph identification. The eye-tracking study contained there is not the work of this thesis' author but still described in section 4.4.3.2 on page 48 as it is highly relevant to this thesis.

> Christin Seifert, Annett Mitschick, Jörg Schlötterer, and Raimund Dachselt. "Focus Paragraph Detection for Online Zero-Effort Queries: Lessons Learned from Eye-Tracking Data." In: *Proceedings of the 2017 Conference on Human Information Interaction and Retrieval*. 2017 (cf. [174])

The contents of chapter 5 on page 67 have been published as mentioned below. The second publication extends the first by an evaluation and discussion of production versus consumption based profiles.

> Christoph Besel, Jörg Schlötterer, and Michael Granitzer. "Inferring Semantic Interest Profiles from Twitter Followees: Does Twitter Know Better than Your Friends?" In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. 2016 (cf. [17])
>
> Christoph Besel, Jörg Schlötterer, and Michael Granitzer. "On the Quality of Semantic Interest Profiles for Onine Social Network Consumers." In: *SIGAPP Appl. Comput. Rev.* 16.3 (2016). ISSN: 1559-6915 (cf. [18])

The investigation of different user interface styles for zero-effort query result presentation in chapter 7 on page 89 has been published as below. The survey is not the work of this thesis' author, but as the results are highly relevant to this thesis, they are presented in section 7.3 on page 94. The results are discussed with technical limitations and its consequences for the implementation as part of this thesis in section 7.4 on page 100.

> Christin Seifert, Jörg Schlötterer, and Michael Granitzer. "User Interface Considerations for Browser-Based Just-in-Time-Retrieval." In: *2015 19th International Conference on Information Visualisation*. 2015 (cf. [176])

Chapter 9 on page 109 has been accepted for publication in the special issue "Interactive Information Visualization" by the Journal of Computer Languages. It unifies the two subsequent publications and extends them by a long-term study on the uptake of QueryCrumbs.

> Jörg Schlötterer, Christin Seifert, Christopher Satchell, and Michael Granitzer. "QueryCrumbs Search History Visualization - Usability, Transparency and Long-term Usage." In: *Journal of Computer Languages* (2020). to appear (cf. [170])
>
> Christin Seifert, Jörg Schlötterer, and Michael Granitzer. "QueryCrumbs: A Compact Visualization for Navigating the Search Query History." In: *2017 21st International Conference Information Visualisation (IV)*. 2017 (cf. [177])

Jörg Schlötterer, Christin Seifert, and Michael Granitzer. "QueryCrumbs for Experts: A Compact Visual Query Support System to Facilitate Insights into Search Engine Internals." In: *2018 22nd International Conference Information Visualisation (IV)*. 2018 (cf. [168])

Section 12.3 on page 162 has been accepted for publication in the reproducibility track of the 42nd European Conference on Information Retrieval. It evaluates various modifications to the random walk strategy in random-walk based network embedding. In a similar direction, we investigated modifications of random-walk based network embedding through representing the graph on multiple abstraction layers in the second publication mentioned. The contents of the latter are described in section 12.4 on page 168.

Fabian Schliski, Jörg Schlötterer, and Michael Granitzer. "Influence of Random Walk Parametrization on Graph Embeddings." In: *Advances in Information Retrieval*. to appear. 2020 (cf. [163])

Jörg Schlötterer, Martin Wehking, Fatemeh Salehi Rizi, and Michael Granitzer. "Investigating Extensions to Random Walk Based Graph Embedding." In: *2019 IEEE International Conference on Cognitive Computing (ICCC)*. 2019 (cf. [172])

The model for learning embeddings from text and network data jointly as described in chapter 13 on page 175 has been published as follows.

Jörg Schlötterer, Christin Seifert, and Michael Granitzer. "On Joint Representation Learning of Network Structure and Document Content." In: *Machine Learning and Knowledge Extraction*. 2017 (cf. [167])

Further publications that are related to, but not part of this thesis are listed below. The first investigates various contextual sensors of mobile phones and how they can be used for zero-effort query generation. The second describes a game with the purpose to provide low level entrance to long-tail resources from the cultural heritage domain and to reveal human search strategies in the long-tail domain.

Jörg Schlötterer, Christin Seifert, Wolfgang Lutz, and Michael Granitzer. "From Context-Aware to Context-Based: Mobile Just-In-Time Retrieval of Cultural Heritage Objects." In: *Advances in Information Retrieval*. 2015 (cf. [169])

Jörg Schlötterer, Christin Seifert, Lisa Wagner, and Michael Granitzer. "A Game with a Purpose to Access Europe's Cultural Treasure." In: *GamifIR@ ECIR*. 2015 (cf. [171])

Part II

FROM CONTEXT TO QUERY

RQ 1

How to transform the user's context into a query that retrieves relevant results?

2 INTRODUCTION

In this chapter, we first elaborate our scenario of zero-effort queries during browsing the Web. We provide an overview of the different levels of contextual granularity within a Web page and how we make use of them to construct zero-effort queries. We then review related work on zero-effort queries with a focus on methods that construct an explicit query. Further we review related work on datasets for zero-effort query construction with a focus on long-tail content.

2.1 SCENARIO

The scenario we address is a user browsing the Web. While the user is browsing the Web, we aim to present additional information beyond the user's current context (i.e., a Web page). To obtain this additional information, we pursue a zero-effort query approach, i.e., we construct queries automatically (and present results accordingly) without explicit user effort. The focus on digital libraries and museum archives as content providers is grounded in the EEXCESS[1] project, in which parts of the work in this thesis were conducted. The vision of this project is to unfold the treasure of cultural, scientific and educational long-tail content for the benefit of all users by taking the content to the user, not the user to the content. The aforementioned long-tail content providers (museums, archives, libraries, ...) typically cannot afford to promote their contents via search-engine optimization and therefore their resources remain hidden in the long-tail of the Web. Neither can they afford to implement dedicated retrieval systems for zero-effort queries, but most of those providers expose their search systems already via APIs. Making use of those existing APIs without altering the retrieval system led to our scenario of zero-effort queries where an explicit query has to be constructed at the client, instead of optimizing the ranking between context and results directly. We further detail this scenario in the following by elaborating on the (active) user information seeking process and context as the available information to replace this active process by an automatic one.

In Broder's model of Web search [26] (see figure 2.1 on the following page), an information need arises within some task. This information need is then verbalized (mentally) and expressed as a query to a search engine. The examination of results is potentially followed by refinement of the query. For the transformation of Broder's model to zero-effort queries, we need to fully automatize the user's steps until

1 http://eexcess.eu – last accessed March 2020

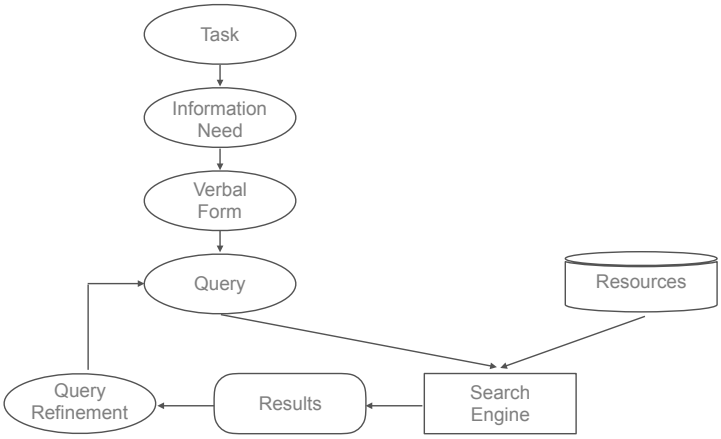


Figure 2.1: Broder's model of Web search [26].

a query is formulated. That is, we need to derive an information need and according query from the context.

2.1.1 *(User) Context*

Context can be defined as a set of descriptive features of the setting, which can be observed (e.g., the current Web page) and/or as a form of engagement with this setting [47]. To make this distinction explicit, the former is referred to as context and the latter as user context. User context emerges dynamically through interaction and the user's background. The dynamic user context in particular is at best partially observable and can hardly be encoded by a set of descriptive features. Still, the user's background can partially be encoded in a user profile, representing the user's interests.

In the Web browsing scenario, the observable context encompasses the Web page itself and corresponding interactions, e.g. mouse clicks or scroll events. We encode the observable *user* context in a user profile, representing the user's interests. This user profile is obtain from past Web pages (i.e., past context). From these two modalities (context and user profile), we aim to derive an information need and further encode this information need as a query to a search engine, hence transforming Broder's model of Web search to zero-effort queries (cf. figure 1.1 on page 4).

A Web page can be composed of different media, such as text, images, video, etc. We focus on textual content only and ignore other media types. The textual content of a Web page can be subdivided on different levels of granularity (from fine grained to coarse): character, term, phrase, paragraph, page. A single character does not provide valuable information on its own and hence we ignore this level of granularity. We detail the remaining contextual granularity levels in section 2.2 on the facing page and explain how we make use of them for zero-effort query generation. On each of these levels, we need to

1) *identify the relevant context* (e.g., identify the paragraph, the user is currently looking at), 2) *formulate an information need* and 3) *express this information need* as a query to the search system. With these three steps, we replace the user's steps to query formulation by zero-effort queries.

2.1.2 *Feature Requirements*

Both, context and user profile require a set of several features to be collected, in order to be used as information source for zero-effort query. In this section, we describe the features we identified as relevant for the transformation of Broder's model of Web search to zero-effort queries.

First and foremost, the textual content of a Web page alongside with its layout needs to be collected as context for the current zero-effort query. Further, interactions with that Web page, such as scroll events or mouse clicks are required to determine the users current focus area. In order to construct a user profile, past context (i.e., previous Web pages) or more generally speaking, the browsing history needs to be collected.

The aforementioned features provide the basis or information source for the generation of zero-effort queries. In order to develop a model that automatically transforms this information source into a query, we further need to collect the user's actual information need and queries a user formulates. Both are relevant, as a query can be ambiguous and not fully representative of the information need. For example, the query "Java" can refer to a programming language, an island or coffee.

To determine the query quality, we further need to collect relevance feedback for the results. This feedback can be gathered either implicitly through result clicks and dwell time on result pages or explicitly by result ratings. Relevance feedback poses a particular challenge in the domain of long-tail recommendations, due to the data sparsity issue: only a few or even no ratings are available for items in the long-tail.

An analysis of available datasets revealed, that the required features are only partly captured in available datasets (cf. section 2.3.2 on page 19), which made it necessary to collect the required data on our own.

2.2 CONTEXTUAL GRANULARITY LEVELS

Figure 2.2 on the following page shows an overview of the different levels of contextual granularity and how they are used in our approach to construct zero-effort queries automatically. Each of the levels *phrase*, *paragraph* and *page* requires the three steps of 1) identification of the relevant context, 2) formulation of an information need and 3)

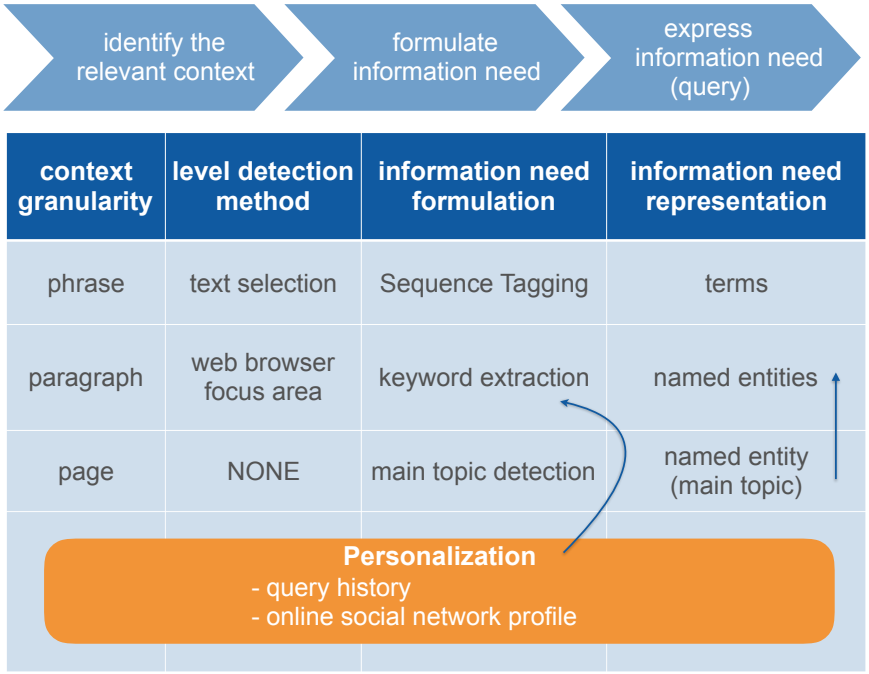


Figure 2.2: Conceptual overview of context granularity levels and corresponding query generation.

expressing this information need as a query to the search system. We omit the *term* level, as it is captured by both, the *phrase* or *paragraph* level, by treating single terms as one-term phrase or paragraph.

On the *phrase* level, we rely on explicit user interaction, i.e., a text selection to identify the relevant context. For the information need formulation on this level, we train a sequence tagging model and the terms identified as relevant by this model form the query. Details on the *phrase* level are given in chapter 3 on page 21.

To identify the relevant context on *paragraph* level (i.e., the paragraph in focus), we first split the page into textual paragraphs and then try to identify the paragraph the user is looking at. The information need on this level is formulated by extracting keywords from the paragraph. The keywords are formed by extracting named entities from the paragraph, which are then combined via a particular query scheme to formulate the actual query. This query scheme consists of a main topic and additional keywords in conjunctive normal form. The evaluation of the automatically constructed queries in this form revealed that the *page* level provides a better source to identify the main topic for the query. Details on the *paragraph* level are given in chapter 4 on page 29.

The *page* level does not require context identification, as the page itself is given as context. It serves as input to identify the main topic for the query construction scheme just mentioned. Details on how we identified the *page* level to provide a better source for the main topic are given in section 4.5.4.2 on page 59.

Beyond the *page* level, a sequence of pages (session or history) can be used to personalize the automatically constructed queries. Instead of utilizing the history of pages, we utilize the history of queries to construct a user profile for personalization. Details of the query history user profile are given in section 4.6 on page 62.

To bootstrap the user profile and enrich it with information beyond what is obtainable from watching the user browse Web pages, we developed an approach to obtain user profiles from online social networks. Typically, user profile construction from social networks is based on content actively posted by users, whereas our approach is applicable to passive users. Passive users do not post content on their own, but express interest via "likes" or following other users. Details of the profile construction from online social networks are given in chapter 5 on page 67.

2.3 RELATED WORK

We present related work on zero-effort queries with a particular focus on automatic query generation. Further we review datasets for zero-effort queries in the context of recommending long-tail resources in terms of their contained features and suitability for the task at hand. As we were not able to identify a publicly available dataset to satisfy all necessary requirements, we collected according data on our own.

2.3.1 *Zero-Effort Queries*

Proactive retrieval, i.e., presenting search results without explicit user interaction was first made popular by Rhodes as Just-in-Time Retrieval [157]. Recently, Rhodes' research has been continued under the topic of zero-effort queries [6] with special emphasis on mobile applications [111]. Zero-effort queries require minimal, ideally no effort of the user in expressing her information need as a query and in obtaining relevant results. While earlier work [28, 120, 156] focused on document retrieval, a wide variety of contents is taken into account in more recent work [184].

Most related work does not create actual queries, but treats the retrieval engine as integral part of the system, which is more similar to recommender systems (see [22] for a survey or [2] for a focus on context-awareness). This observation was also made by Hagen et al. [66], who use keyqueries [59] to retrieve related documents. The keyquery approach is not applicable in an online setting, as it requires issuing several queries to a reference search engine to determine the keyquery. At least 3 queries have to be sent to this search engine and for a black box search engine, even with up to 128 queries, the authors achieved a success rate of 60% at best (i.e., a keyquery could be found). Similar to keyqueries, Dasdan et al. [42] generate query signatures

for documents. These query signatures are constructed from the least frequent or randomly sampled terms in the document and sent to a reference search engine in order to find a near-duplicate document.

As one of the few approaches for automatic query generation, Nascimento et al. [137] obtain query candidates from a document via n-grams and noun phrases and rank these candidates based on the term frequency and text position. They retrieve results for the top-n query candidates and re-rank the results by matching them against the source document (i.e., their approach is limited to documents). Similarly, Boley et al. [23] use the intersection of terms with the highest text frequency (word frequency within a document) and highest document frequency (word frequency across documents) out of a cluster of documents as search query. The presence of a corpus for calculating document frequency is also a pre-requisite in the approach of Yang et al. [222]. They used the BlogScope (a social media monitoring platform) corpus and ranked noun phrases as candidates by tf-idf scores or mutual information of the terms in candidate phrases. Ranking by tf-idf scores of individual terms in candidate phrases requires to sum over the term scores to determine the phrase score. This summation naturally favors longer phrases, an issue that was addressed by Sangaralingam et al. [89]. Instead of summing individual term tf-idf scores, they directly calculate a pf-idf (phrase frequency - inverse document frequency) score for the candidate phrases.

Ševcech and Bieliková [180] lift user annotations to identify interesting parts of a document and represent them as a graph. Using spreading activation on this graph, they identify the most important terms to form a query. Golovinsky et al. [60] also used annotations to query related resources. Both methods require the existence of annotations, i.e., manual user effort.

Most similar to our work, Lee and Croft [109] extract chunks, i.e., noun phrases and named entities, from user selected text segments. They estimate chunk importance with a learned Conditional Random Field (CRF) Model. Training their CRF requires a measure of retrieval performance for individual chunk configurations, which is typically not available for long-tail resources (cf. next section 2.3.2 on the facing page and section 3.3 on page 26). Cheng et al. [38] suggest queries, based on browsing history and corresponding search history logs (i.e., when browsing was followed by a query). This approach also suffers the data-sparsity issue in the context of long-tail resources.

As mentioned before, most of the works still focus on a certain dimension of retrieval or treat the retrieval system as an integral part of the application. This setting is also reflected in publicly available datasets: They are highly specific in the retrieval task, e.g. in TREC contextual suggestion track[2] user profiles and context are combined to suggest attractions, such as restaurants or museums. Yet, a lot

2 https://sites.google.com/site/treccontext/ – last accessed March 2020

of information retrieval systems with a wide variety of content are readily available, exposing their contents via public APIs (e.g. Bing Search API[3]). This is particularly true for digital libraries, featuring high quality content and rich metadata. There is a lack of datasets, that allow to develop and evaluate proactive query construction strategies, making use of these readily available repositories.

2.3.2 *Datasets*

Providing long-tail recommendations is a highly challenging task, first of all because of the data sparsity issue: only a few or even no ratings are available for items in the long-tail. To overcome this problem, the authors in [143] partition the whole item set into head and tail parts and cluster the items in the tail. In [110] recommendations are obtained by combining the items in a user's personal long-tail with users, which have those items in their head portion. While these approaches still require the existence of at least a few ratings in the tail or even the existence of dense data in the head, Stickroth et al. [192] aim to provide high quality recommendations in a network with a small amount of users and items (and hence without the presence of a dense head). Therefore the authors propose a multilevel approach, with a decreasing degree of personalization and different recommendation strategies at each level. Their dataset encompasses 60 ratings on 151 items by 175 users and is not published. Closest to our work, Wang et al. [211] conducted a user study in the cultural heritage domain in which they elicited user models with ratings of museum objects of the Rijksmuseum Amsterdam from 39 participants.

Most of the approaches for user data collection for long-tail domains use server-side data logging. A representative example is the smartmuseum approach were user interests are either manually given or by tagging and rating of resources [160]. A game-based approach to server side collection was pursued by Wang et al. [211] who used an interactive quiz to collect ratings for museum objects. Goecks and Shavlik [58] use client-side data collection in a Web browser for user interest detection based on the text of the Web page, clicked hyperlinks, scrolling and mouse activity.

All of those datasets capture features that are relevant for zero-effort queries only partly. To the best of our knowledge, there is no publicly available dataset, which accounts for the specific challenges of long-tail recommendations and contains the required data.

3 https://azure.microsoft.com/en-us/services/cognitive-services/bing-web-search-api/ – last accessed March 2020

3 PHRASE LEVEL

This chapter describes the query generation process on the contextual granularity of a phrase or sequence of terms.

The most accurate way to identify the phrase currently read by the user is eye tracking (and even eye tracking accuracy is not perfect). Browser events, such as mouse movements or scroll position yield only limited accuracy [69]. Thus, for the identification of the relevant context, we rely on explicit user interaction on this level, i.e., a text selection, which is a strong indicator for reading focus.

In a user study, analyzing how users search for related material and collecting a dataset for personalized zero-effort queries, we discovered that the majority of queries users issued for a particular text selection contain one or more terms from the selected text [175]. That means, the relevant information for the query formulation is already there and we need to find a way to extract it. We first provide a brief description of the study with focus on the relevant parts for this chapter in section 3.1, followed by the analysis of text selections and corresponding queries in section 3.2 on page 23. We conclude the chapter with our approach to extract the relevant terms to formulate a query from a text selection in section 3.3 on page 26

3.1 METHODOLOGY

We first aimed to gain a deeper understanding, how *users* formulate queries within a given context and further use this knowledge to formulate queries automatically. As mentioned in section 2.1.2 on page 15, to the best of our knowledge, no publicly available dataset contains all the relevant features (cf. also section 2.3.2 on page 19) to foster this analysis. Therefore, we conducted a user study, in which we collected the features identified as relevant. The goal of this user study was two-fold: First, we aimed to gain a deeper understanding on the *user* query formulation process as just mentioned and second, we aimed to use the collected data for zero-effort query construction.

For the user study, we developed a web browser extension and defined tasks for users to solve by using this extension. In the defined tasks, we distinguish between two basic web usage scenarios: content consumption and content creation (e.g., writing a blog entry). The latter is not relevant to this chapter, hence we only report about the content consumption tasks. Particularly relevant for this chapter is the task to "annotate a Web page", which was predefined by us. In this content consumption task, we asked users to annotate Web pages

with additional relevant resources. Following the assumption that resources are relevant for fragments of Web pages only [118], we used annotations to identify the mapping between a selected text fragment of the Web page and the most relevant resource for this fragment. After selecting a text fragment, users had to issue a query to a search engine in order to find additional resources and annotate the text fragment with relevant resources found. Users also were asked to fill out a questionnaire to collect privacy related features and demographic data.

3.1.1 *Procedure and Participants*

The study starts with a general introduction including a demonstration of the interface, followed by explanations about the tasks and privacy issues. Then participants perform the study by executing a combination of predefined tasks (our content consumption and creation tasks) and arbitrary Web tasks (e.g., watching a video), alternating between those two. For each task participants indicate task specific information, like the task label, their experience level and associated topics. During task execution all interactions are logged. Finally, participants fill out the post-study questionnaire.

8 German speaking students participated in the study, 5 male and 3 female. The average age was 27 years, ranging from 21 to 34 years. 2 participants considered themselves as computer experts, 6 as average computer users. 7 participants stated that they were heavy computer and Internet users with a usage of more than 2 hours daily, one participant used the Internet and computer daily, but not more than 2 hours per day.

3.1.2 *Prototype*

We implemented the prototype as an extension to Google's Chrome browser. As backend for search queries, we use the Europeana API[1]. Figure 3.1 on the facing page shows a screenshot of the extension, which is injected in every Web page, simulating the behavior of a sidebar. The user can switch the injection on or off respectively by clicking the extension icon (01). On top of the sidebar the user can define queries (02) and select the preferred language of the results (02a). The retrieved results are displayed as a list, with a short summary for each result item (03). Clicking on a result retrieves a detail view, which is shown as an overlay on the current page. Rating icons are available for each resource (04) and the source URL of the resource can be retrieved by the button at 05. Already existing annotations are highlighted in the text (06). For a current text selection, comments

1 https://pro.europeana.eu/page/search – last accessed March 2020

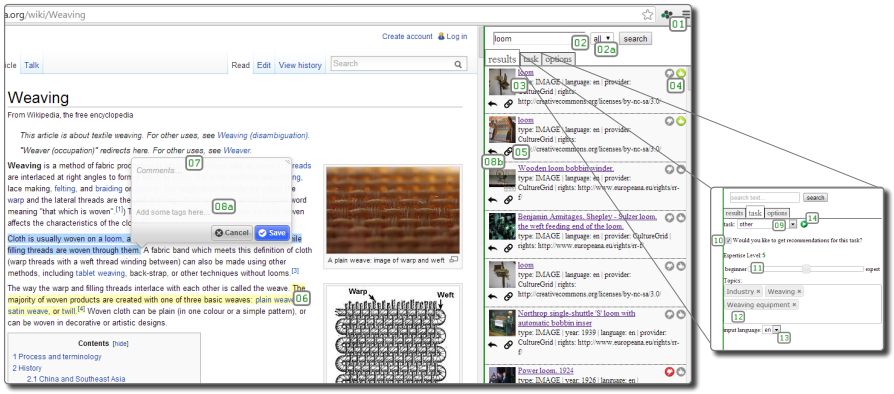


Figure 3.1: Screenshot of the test data acquisition user interface. Normal browser window content (left, Wikipedia page on weaving), browser extension (right pane), available settings of the option panel (right, center).

(07) and the respective resource URI (08a) can be added. The URI is automatically copied to the annotation's interface when icon 08b is clicked.

The smaller window in figure 3.1 shows the content of the task tab. Here, the user can select the task type ("annotate a Web page", "write a blog entry" or "other") at 09. For the latter, the user is asked to provide a custom label after task execution, and an additional check box (10) is shown to indicate, whether automatic recommendations would be desirable for this task. The user can adjust her level of expertise on the task at hand with the slider at 11 on a scale from 0 (lowest) to 10 (highest). The topics related to the specified task are defined at 12. This input field features auto-completion for DBpedia-categories, i.e., the user gets suggested DBpedia-categories that possibly match her input while typing. The language for auto-completion can be selected at 13 (supported languages: French, English and German). After a task has been started (14), it is not possible to change its type anymore, while all other settings may still be adjusted, because they might not be known in advance. For instance, the topics related to a task are usually discovered during its execution and the task can be labeled more meaningful.

3.2 ANALYSIS

In this section we provide a qualitative and quantitative analysis of the data we collected during the user study with a focus on the relation between the selected text and the query issued for the selection (cf. section 3.2.2 on the next page).

3.2.1 *Dataset Statistics*

Table 3.1 provides an overall summary of the collected data. Column

Table 3.1: Dataset overview. Time in minutes.

users	tasks	annotations	queries	views	ratings	time
8	217	1332	4562	3267	15043	10252

"views" refers to the number of times users opened the detail page of results. We count each single view and do not distinguish between different results or one result that has been opened multiple times. Column "time" accumulates the duration (in minutes) of all performed tasks, "annotations" counts the total number of users' annotations of Web pages with any of the retrieved results. Interestingly, the amount of ratings is more than three times larger than the amount of result views. It is obvious that a lot of results have been rated solely by the short summary as provided in the result list. A potential reason for this is that users assessed the quality of additional information provided in the detailed view as below average in the questionnaire (∅2.75 on a 1-5 scale).

We collected 8,091 positive and 6,826 negative ratings in the predefined tasks, amounting to 14,917 ratings. The difference to the amount of ratings in table 3.1 stems from ratings users provided in their arbitrary chosen tasks. The high amount of ratings is explained by the assignment asking participants to rate at least 10 results for each search. On average, users rated equally positive and negative, we found no significant difference between the number of positive and negative ratings (Shapiro-Wilks test for normality, $W = 0.9305, p = 0.2777$ for negative, $W = 0.9608, p = 0.7063$ for positive ratings, paired T-Test at confidence level $\alpha = 0.05$: $t = 1.5687, df = 14, p = 0.139$).

3.2.2 *Selections and Queries*

We analyzed the distribution of query terms and selected texts for the content consumption tasks (there was no text selection in the content creation setting). Generally, users preferred to issue queries containing only few terms. The majority of the queries (81%) contains one to three terms with an average query length of 2.8 terms. These numbers are similar to findings from general search engine usage [195]. The text selections were larger, with an average of 8.3 terms. The majority of the selections contains 1 to 4 terms (60%). 10% of the selections had more than 25 terms. 24 ($\approx$ 1%) selections had more than 60 terms.

Figure 3.2 on the facing page shows the relation of query terms and terms in the selected text. For 38% of the queries, there was no overlap between the selection and query terms, i.e., no term of the selection

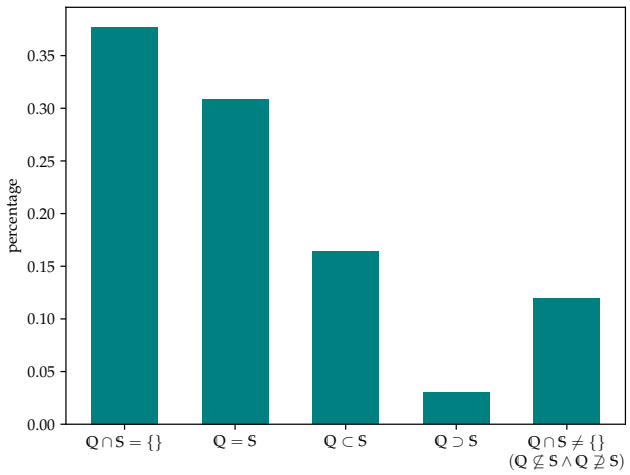


Figure 3.2: Overlap between query terms (Q) and selection terms (S).

was contained in the query. The majority of queries (62%) shows at least a partial overlap with the selection. In particular, the query term set was equal to the selection set in 31% of the cases. In 16% the query term set was a true subset of the selection, i.e., the query has been formulated only by terms contained in the selection. 3% of the queries have been an expansion of the selection by additional terms (selection terms were a true subset of the query terms). The final 12% are queries that have been formulated by selecting some terms of the selection and expanding by additional terms (partial overlap).

The high overlap between query and selection terms becomes even more evident when looking at the vocabulary of selections and queries as depicted in figure 3.3. The majority of the query vocabulary (approx-

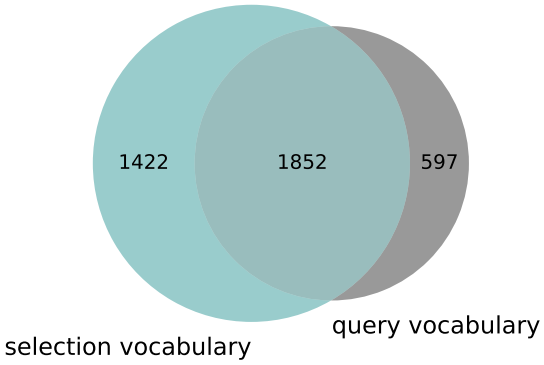


Figure 3.3: Vocabulary overlap between queries and selections.

imately 72%) overlaps with the selection vocabulary, i.e., these terms have been present in the selections. Expectably, stemming further increases the vocabulary overlap to approximately 76% and decreases the fraction of queries where query and selection term set are disjoint to 36%. However, as we do not have information about particular search engine internals (they are a black box to us), stemming might

interfere with the query processing of the search engine. In the analysis, we applied a Snowball stemmer, specific to the language of the page in which the text was selected, whereas the search engine might for example apply a Porter stemmer. Hence we focus on text selections and queries without pre-processing, out of which 62% show an overlap on syntactic level.

The numbers reported here differ from the original paper, where we only identified 15% of the queries having no overlap with the selection at all [175]. This is due to an error in the analysis of the original paper: We only analyzed the data from 6 users as we had not yet received the data from all 8 participants. Apparently, the behavior of those last two users was different from the others, as most of their queries had no overlap with the selection. These two users are included in the analysis presented here, hence the fraction of queries without an overlap on the text selection increases from 15% to 38%. Still, in the majority of queries (62%) we observe at least a partial overlap with the text selection. The evaluation of learned queries as presented in the next section has been carried out on the full set of 8 users in the original paper as well.

The analysis of text selection and corresponding query indicates that in 38% of the cases there is no *syntactic* overlap between the selected text and the query terms and thus, for this cases it is impossible to derive any information for the query directly from the text. Then again, in the majority of cases, information about which terms to use for a query is already contained in the selected text. The selected text containing readily usable information on a syntactic level motivates our approach to extract this information from the text selection as presented in the next section.

3.3 LEARNING QUERIES

Given the results of the previous section, we try to infer, which terms of the selection should actually be used for a query by labeling them as either relevant or not relevant. To predict the relevant terms, we applied a linear-chain conditional random field model (CRF) [106]. This model predicts a sequence of labels for a given input sequence, conditioned on the input features, i.e., it models the probability of the output variables (the labels), conditioned on the observed variables (the input features). We opted for a CRF, as it is advantageous over a Hidden Markov Model in efficiently accounting for dependencies among input features.

For the evaluation, we considered only those text selections, in which at least one term was contained in the corresponding query, amounting to 2449 sequence-query pairs. The input features used were the identity of the term (i), i.e., the term itself, whether the term starts with upper- or lowercase (c), an indicator if the term is a stop word

(s), the part-of-speech (POS) tag (t), and the equivalent input features of the preceding (s_{-1}, c_{-1}, t_{-1}) and succeeding (s_{+1}, c_{+1}, t_{+1}) term. A single sample in the training set therefore consists of a text selection enriched with the aforementioned features and a label for each term of the selection, which indicates if the term is also contained in the corresponding query (and hence considered relevant).

The list of stop words is the one provided by the "tm" package for R[2], the POS tags were obtained with NLTK [20] and the CRF models were computed with Mallet [128].

3.3.1 *Results & Discussion*

We evaluated the performance of 29 feature combinations using 10-fold cross-validation. In order to evaluate the stability across users and tasks we also performed cross-validation on splits defined by users (all but one user as training and one user for test), and tasks respectively.

The best performing feature combinations are shown in table 3.2. As

Table 3.2: Accuracies [%] for query prediction from selected text. Cross-validated using splits over users, tasks, and 10-fold random.

		feature set			trivial	
		i, c, t	i, t	c, t	rejector	acceptor
users	mean	76	77	75	51	49
	SD	15	15	18	35	35
tasks	mean	82	83	82	71	29
	SD	6	6	7	8	8
10-fold	mean	89	88	84	71	29
	SD	1	2	1	2	2

i - the identity of a term, i.e., the term itself
c - whether the term begins with upper- or lowercase
t - POS tag

the CRF model assigns a label to each term in the selection (identifying it as relevant or not relevant), accuracy refers to the ratio of correctly labeled terms to the total number of terms. Incorporating a term itself as a feature (i, c, t & i, t) leads to the best results, but may not generalize well due to the limited vocabulary in the dataset. Nevertheless, feature combinations without the words provide similar results as well (e.g., the combination of case-identifier and POS-tag, c, t) and thus are the better option.

The standard deviations reveal, that the query behavior is stable over tasks, but not over users. In fact half of the users incorporated

2 http://cran.r-project.org/web/packages/tm/ – last accessed March 2020

the major part of the selection into their queries and the queries of the other half contained only a minority of the selection terms. Thus, prediction performance drops for the evaluation over users.

3.4 CONCLUSION

The analysis of the dataset revealed that users tend to use terms of the selection when constructing a query for additional resources related to that text selection. In 62% of the selection/query pairs we observed at least a partial overlap between text selection and query. That means, the information of which terms to use for a query is (partially) available already on syntactic level but needs to be extracted. With a CRF-model, we were able to extract the relevant terms with up to almost 90% accuracy in a supervised learning setting. Still, this high accuracy does not guarantee a good query, but a query that is similar to a query a human would issue.

4 PARAGRAPH LEVEL

The query generation process on paragraph level consists of two steps: First, the identification of the relevant paragraph (i.e., the paragraph the user is currently reading) and second the generation of a corresponding query. The identification of the relevant paragraph again subdivides into a two-step process: First, the whole page has to be split into distinct paragraphs and second, the paragraph in focus has to be identified. We refer to the former as paragraph extraction and to the latter as (focus) paragraph detection.

We start this chapter by introducing the methodology we applied for query generation on paragraph level in section 4.1. In short, we conducted a user study with our methods towards query generation. We provide a detailed description and evaluation (based on the collected data) of these methods for the individual steps of identifying the focus paragraph in section 4.3 on page 39 (extraction) and section 4.4 on page 46 (detection) and generating a corresponding query in section 4.5 on page 52. We finish this chapter by showing how the generated queries can be personalized in section 4.6 on page 62 and present an approach to bootstrap personalization in the following chapter 5 on page 67.

4.1 METHODOLOGY

Similar to the phrase level (cf. chapter 3 on page 21), we conducted another user study, building on the findings of the previous chapter. These findings mainly comprise the observation that information on how to formulate queries is at least partially available within the context even on syntactic level and, if this is the case, *user* formulated queries can be learned with high accuracy. Still those queries are not necessarily *good* queries. Further, in the previous chapter, we relied on a text selection as explicit indicator for context identification, whereas in this chapter we aim for an implicit detection. Therefore, the goal of the user study is again two-fold: First, we aim to evaluate our approaches for context detection and query formulation, which we developed based on the findings of the previous chapter. Second, the collected data serves to evaluate further modifications and enhancements.

We collected data for evaluating context driven zero-effort queries in a setting where the search engine is a black box. The data has been generated by 77 users, containing the users' paths through Web pages and 656 queries for additional material related to textual paragraphs

from these pages. 8528 ratings are provided for the results retrieved, alongside with the users' actual search intentions.

4.1.1 *Conceptual Overview*

The data collected during the user study covers different stages of the user information seeking model of Marchionini & White [126], as shown in figure 4.1. We omit the steps of recognizing an information

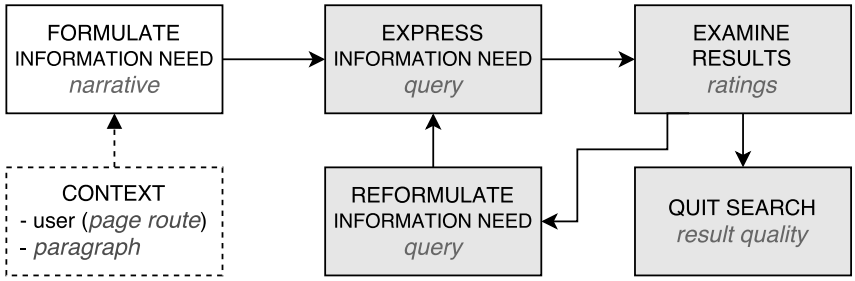


Figure 4.1: Conceptual overview of data contained in the dataset, based on a user information seeking model [126]. Grey boxes indicate fully observable information, white boxes indicate information, which is only partially observable. The actual representation is indicated via the italic terms.

need and accepting the challenge to fulfill this information need from the original model as those are inherent in our setting. Instead, we start with the creation of a mental model of the information need. Thus, our first step is the *formulation* of the information need in the user's mind, which is *expressed* (as a query) in the second step. The *examination* of results in the third step is potentially followed by *reformulating* the expression of the information need and finally, the examination *stops* in the fourth step. We explicate the *contextual influence* on the information need *formulation* and *expression*. Following Dourish [47], we distinguish between context as a set of descriptive features of the setting, which can be observed (i.e., the paragraph in focus) and the user context, which is rather a form of engagement with this setting. Obviously, the latter is not directly observable, but at least partially encoded in the user's past interactions (i.e., the route of pages, reflecting a user's interests).

More formally, the dataset comprises a set of users U and a route of pages $< w_i, ..., w_j >$ for each user u through the set of available pages W. The whole content of each page visited on the route is also available from the dataset. The combination of a predefined paragraph $p \in P$ on page $w \in W$ and user $u \in U$ forms an information need $i \in I$, which is expressed as query $q \in Q$. The query q yields a list of results $r =< r_m, ..., r_n >$ from all results R in the repository, which are judged via relevance feedback F, depending on user, paragraph, query and information need. The results are stored including a rich

set of metadata. In summary, the data contains:
- the route of pages $< w_i, ..., w_j >$ per user u
- the query process $< p, u > \xrightarrow{i} q \rightarrow r$ per paragraph p
- relevance feedback $F(u, p, i, q, r)$

Figure 4.2 illustrates the formalization with potential routes through pages (top), and the query and rating process for a single page or paragraph (bottom).

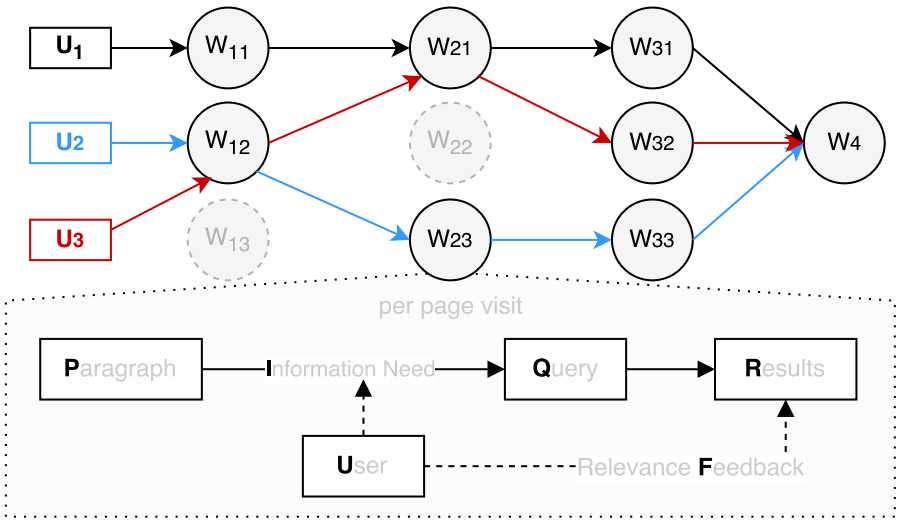


Figure 4.2: Potential routes through the set of predefined pages (upper part) and query & rating process for a paragraph on a single page (lower part).

4.1.2 *Acquisition Methodology*

We developed a browser extension to collect the data and defined tasks for users to solve by using this extension. The participating users were recruited via an online recruitment system of the university and were paid ten Euros for taking part in the study, which lasted approximately an hour. The data collection via the browser extension was accompanied by a questionnaire, in which users were asked to indicate the result quality per task, their search intention for each task and demographic data. Further, participants were asked to provide their assessment on the difficulty of finding relevant results, as well as on the helpfulness, extensiveness and quality of the retrieved results using a five point Likert scale (1 - strongly disagree, ... , 5 - strongly agree). The questionnaire concluded with a set of open questions, such as "what could have been improved?", "what did you like?", etc.

4.1.2.1 *Participants*

77 university students from different disciplines took part in the study. Among the most prominent study paths of the participants were

business administration and economics (19) and law (11), followed by teaching (8) and political science (8). The average age was 23, ranging from 19 to 31 and 46 of the users were female, 31 male. 67 considered themselves as average computer users, 9 as experts and 1 did not provide this information. All of them use the Internet on a daily basis, with 28 using it less than 2 hours per day and 49 using it more than 2 hours per day.

4.1.2.2 *Procedure*

The data acquisition started with a general introduction, explaining the aim of the experiment. After the introduction, the procedure was explained with the help of a screencast. Since the general process is the same for each task, an exemplary task was performed in the screencast. Then, participants performed four tasks on their own.

4.1.2.3 *Prototype*

A screenshot of the prototype used to acquire the data is depicted in figure 4.3. First, users had to indicate the relevant part of text

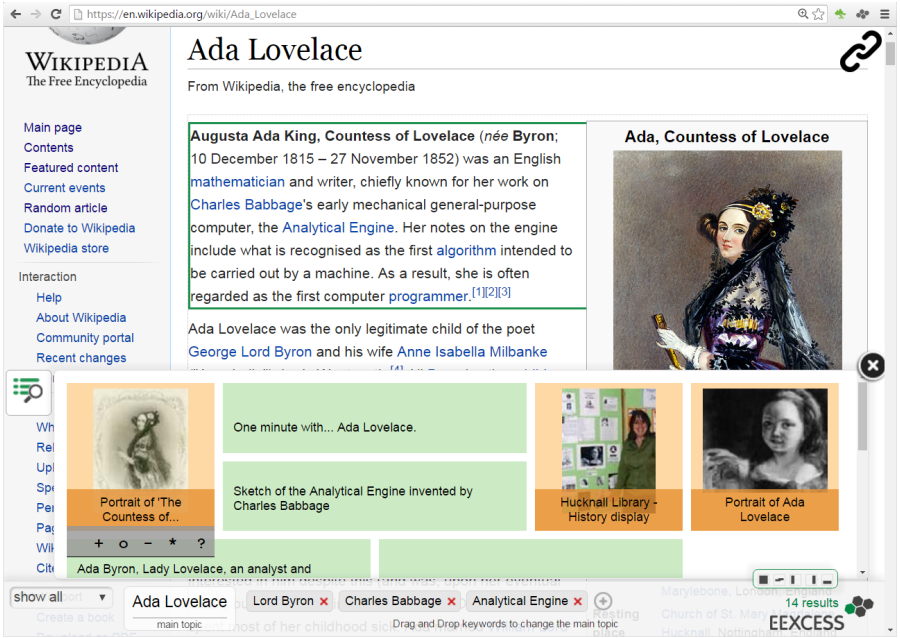


Figure 4.3: Screenshot of the prototype used to collect the data. The paragraph framed in green indicates the relevant part of text within the page, the query is located at the bottom of the page and the results are shown in a popup.

within the page, i.e., the focus paragraph, framed green in the figure. To this end, the paragraphs extracted by our algorithm are framed with a dotted gray border (details in section 4.3 on page 39) and the paragraph deemed in focus by our algorithm is framed green (details in section 4.4 on page 46). Users could change the focus paragraph by

simply clicking on a different paragraph framed with a dotted gray border. If no suitable paragraph had been extracted by our algorithm, users were instructed to select the relevant piece of text and mark that as focus paragraph. Then an automatic query was generated for the relevant paragraph, consisting of a main topic and additional keywords as shown at the bottom of figure 4.3 on the preceding page. We omit the details of the automatic query construction here, as they are not relevant for the data collection and refer the reader to section 4.5 on page 52 for a detailed explanation of the approach. A popup shows the search engine results page (SERP) and users could navigate to the actual result by hovering over it and clicking the question mark at the bottom right of it (visible for the first result in figure 4.3 on the preceding page). Users were instructed to inspect the results of the automatic query and provide relevance feedback. Rating markers are shown for the first result in figure 4.3 on the facing page. Afterwards, they were asked to adapt the query, in order to retrieve better results. To adapt the query, users could add and remove keywords, set a keyword as main topic via drag and drop or edit the main topic. In addition, a filter menu was provided left to the query terms. It contained an option to filter the keywords for persons or locations and another option to restrict the set of keywords to a particular aspect of the paragraph in focus (i.e., a sub-paragraph).

Different versions of the prototype were provided to the participants, that varied in feature-richness. Additional features comprised a *visual explanation* of the suggested query terms (and corresponding results) on user interface level and the *personalization* of suggested query terms. To provide a *visual explanation* of the query terms, the origin of a term was highlighted in the selected paragraph of the page whenever a user hovered over a query term. Also, corresponding results, i.e., this term occurred in their title, were highlighted. We provide a detailed explanation in chapter 8 on page 103. The *personalization* restricts the suggested query terms to a particular aspect (i.e., a single sub-paragraph) of the selected section. This means a pre-selection is carried out for the second query filter mentioned before. The query term filter is pre-selected for the aspect (sub-paragraph) that has the highest overlap with the user profile. The construction of this user profile is based on previous queries, see details in section 4.6 on page 62.

4.1.2.4 *Tasks & Test Material*

The general procedure was the same for each task: First, users had to navigate to a particular page and a predefined paragraph within that page. For this paragraph, a search query was generated automatically (cf. section 4.5 on page 52) and users had to rate the results. Finally, they were instructed to adapt the query (and rate the corresponding results), until one of the following criteria, indicating the result quality, was met: (i) they are satisfied with the results, (ii) it is not possible to

retrieve relevant results or (iii) the time is over. The timing bounds were determined in a pretest with a small user group and set to 8 minutes per task.

Users were free to choose a Wikipedia page from a predefined list in the first three tasks. To gather this list, a featured article from each category was chosen at random from the list of Wikipedia's featured articles[1]. The resulting list contained 35 Wikipedia pages per task from which the users could chose. On the chosen page, users were asked to navigate to a particular section: the third in the first task, the second in the second task and the fourth in the third task. The section was determined by the order of sections within the table of contents. In the fourth task, the Wikipedia page was predefined and the same for all users. In this task, users were instructed to navigate to the introductory section.

4.1.2.5 *Search Backend*

The search results were retrieved from the REST-APIs of Mendeley[2] and Europeana[3]. While the retrievable contents in Mendeley comprise scientific documents, Europeana features cultural objects in a variety of formats, such as text, audio, video and 3D. Hence, with Mendeley we perform traditional document retrieval, while with Europeana, the search is performed over the metadata of cultural objects and therefore we cover different possible resource representations in digital libraries. Further these providers cover two different domains: science and cultural heritage. Both providers are not limited to a particular area within their domain, opposed to for example PubMed[4] (medicine) or dblp[5] (computer science). As an aggregator, publishing data from institutions all across Europe[6], Europeana further has a large coverage. In the case of Mendeley, the query terms were sent to the API as a simple keyword list, while for Europeana a boolean query was constructed, following a specific scheme defined in section 4.5.1 on page 54. From each provider, at most the top-10 results were shown to the user, interleaved via Round-Robin.

1 https://en.wikipedia.org/w/index.php?title=Wikipedia:Featured_articles&oldid=699545245 page revision used to create the list – last accessed March 2020
2 http://dev.mendeley.com – last accessed March 2020
3 https://pro.europeana.eu/page/search – last accessed March 2020
4 https://www.ncbi.nlm.nih.gov/pubmed/ – last accessed March 2020
5 https://dblp.uni-trier.de – last accessed March 2020
6 https://www.europeana.eu/portal/en/explore/sources.html – last accessed March 2020

4.1.3 *Dataset Statistics*

During the study, 656 queries (automatic and user generated) were issued in 272 tasks[7], resulting in 2.4 queries on average per task. Details have been viewed for less than 4% of the results (331), which indicates that the results have been rated based on the result surrogates, rather than the results itself. Table 4.1 shows the result quality per task. The

Table 4.1: Result quality.

task	satisfying results	no results	timeout	missing
1	9	11	51	6
2	19	16	35	7
3	28	16	24	9
4	21	5	34	17
total	77	48	144	39

column *satisfying results* indicates, that the user was fully satisfied with the retrieved results and therefore, no further adaption of the query was required. The column *no results* describes the situation when the user assumes, that the search system is not able to provide any relevant results, no matter how she formulates the query (i.e., she assumes the database does not contain relevant content). The third column counts the occurrence of timeouts, i.e., not meeting one of the two aforementioned criteria within 8 minutes. The last column indicates missing values, where users did not provide a result quality assessment. We attribute the increased frequency of timeouts in the first task to the required time to get familiar with the software and to not having an intuition yet on how long it takes to execute the task.

On average, 13 results have been rated per query. Table 4.2 on the following page shows the ratings for different user groups. The ratings were based on the question, whether users consider a result as potentially helpful additional information. Possible values were "not relevant" (-), "cannot judge" (0), e.g., due to lack of domain knowledge, "relevant" (+) and "perfect match" (*). The latter indicated a single result, which would already fully satisfy the information need of the user.

The column "avg score" provides the average rating score for which results considered as "not relevant" are weighted by -1, "cannot judge" by 0 and both, "relevant" and "perfect match" by 1. As can be seen from the table, users with the personalization feature tend to rate more positively, with a total average score of -0.16 for the personal-

7 We only count queries, for which results have been rated. Some users did not rate a single query for some tasks, hence the number of tasks does not match the expected total (272 vs 308).

Table 4.2: Relevance feedback.

#users	e	p	-	o	+	*	total	avg score
24	1	1	1086	377	610	123	2196	-0.16
20	1	0	1112	416	479	124	2131	-0.24
17	0	0	1076	383	532	97	2088	-0.21
16	0	1	1062	321	608	122	2113	-0.16
77			4336	1497	2229	466	8528	

e - whether the explanation feature was activated (1) or not (0)
p - whether the personalization feature was activated (1) or not (0)

ization group and -0.23 for the non-personalization group. However, the score as provided in the table includes all queries, whereas the personalization feature has hardly to no effect both on queries issued in the beginning and queries modified by the user. Obviously, more results have been rated as irrelevant than relevant.

The user provided assessment as listed in table 4.3 on the difficulty of finding relevant results and result quality shows a similar picture in terms of perceived result quality. The table shows the average answer

Table 4.3: Subjective assessment.

#users	e	p	easy	helpful	extensive	quality
24	1	1	2.50	2.75	3.75	2.75
20	1	0	2.35	2.65	3.30	2.30
17	0	0	2.65	2.88	3.53	2.76
16	0	1	2.13	2.50	2.94	2.63

e - whether the explanation feature was activated (1) or not (0)
p - whether the personalization feature was activated (1) or not (0)

values to the following four questions:

1. How easy was it to find relevant results (column "easy")
2. How helpful were the results (column "helpful")
3. How do you rate the extensiveness of results (column "extensive")
4. How do you rate the quality of results (column "quality")

The possible answer values ranged from 1 (very difficult, not helpful at all and very low) to 5 (very easy, very helpful and very high). Except for extensiveness, all values are below the theoretical average of 3.0: While users assess the amount of (potential) results above average, they asses their helpfulness and quality as below average and rather difficult to find.

4.1.4 *Data Cleansing*

From the collected data, we removed all paragraphs and their associated queries that could not clearly be assigned to a task (i.e., queries executed on pages other than the predefined ones). Also we removed all paragraphs and their associated queries, in which the first rated query was an already adapted one and not the automatically generated query. Furthermore, we removed queries, were no rating has been provided for any of the results. The cleansing was performed in order to ensure that the query sequences associated to a paragraph start with an initial automatic query and all subsequent queries are modifications by the user. Also, the cleaned data contains only queries and corresponding results for which ratings have been provided, since those are required to determine the query performance. The cleaned dataset consists of 251 paragraphs and 558 associated queries, executed by 69 users in 228 tasks. The reason why there are more paragraphs than tasks is that users could modify the automatic pre-selection of a paragraph in a task. Even though they were instructed to apply this modification before executing any query, some users seem to have modified it after they already executed queries and rated the results. 6985 relevance ratings have been provided for the retrieved results. All subsequent evaluations are carried out on the cleaned dataset if not mentioned otherwise.

4.2 LEARNING QUERIES REVISITED

In section 3.3 on page 26 we showed that a linear-chain conditional random field model (CRF) can predict which terms of a text selection to use for a *user-generated query* with almost 90% accuracy. However, this evaluation had two limitations: First, we considered only selection-query pairs, in which at least a partial syntactic overlap between selection and query was present. Second and more important, a *user-generated query* does not necessarily constitute a *good query*.

In this section, we re-evaluate the CRF model on queries that can be considered as *good queries*. Therefore, we consider all queries, for which the average rating of the corresponding results is at least as high as the average rating of result sets assessed as "satisfying results". These are the result sets returned by the last query in a task that has been ended due to the user being satisfied with the retrieved results (cf. table 4.1 on page 35). This definition of *good queries* resulted in 211 paragraph-query pairs obtained from the dataset before cleansing (cf. section 4.1.4). We chose to use the data before cleansing in order to increase the amount of pairs. For this evaluation, only the ratings of a query's results and the assignment of the query to a paragraph are necessary. Other properties, such as for example the first query for a paragraph being an automatic query, are irrelevant, therefore we can

work with the unfiltered data. Contrary to the evaluation in section 3.3 on page 26, this setup can also include paragraph-query pairs without any syntactic overlap between paragraph and query.

For training the CRF model, we follow the same setup as in section 3.3 on page 26. We treat the query construction as a sequence labeling task: Given the sequence of terms in the paragraph, we try to infer for each term, whether it should be used in the query. The CRF predicts a sequence of output labels (relevant/not relevant) for a given input sequence, conditioned on the input features. The input features used were the term itself, its part-of-speech (POS) tag and whether the term starts with an upper- or lowercase letter. The POS-tags were obtained with NLTK [20] and the CRF models were trained with Mallet [128].

4.2.1 *Results & Discussion*

The performance was evaluated over a 10-fold cross-validation with the results provided in table 4.4. All values are macro-averaged.

Table 4.4: CRF results for *good queries*.

features	accuracy	trivial rejector	precision	recall
i, c, t	0.90	0.85	0.64	0.44
c, t	0.85	0.85	0.42	0.10

i - the identity of a term, i.e., the term itself
c - whether the term begins with upper- or lowercase
t - POS tag

The accuracies of both feature sets (i, c, t and c, t) of the current evaluation are close to the accuracies in our previous evaluation (0.89 and 0.84 - cf. section 3.3.1 on page 27). However, this time a trivial rejector results in an accuracy of 0.85, whereas it previously resulted in an accuracy of 0.71. This is due to the fact that a paragraph typically contains more terms than a manual text selection and consequently more terms that are irrelevant for the construction of a query.

Same as in the previous evaluation, the vocabulary of our dataset is rather limited, hence the bottom row (feature set c, t) is of higher interest to us. While the trivial rejector is on par with the CRF model in terms of accuracy, it would not provide any information for query construction. Still, the CRF model also provides little information for query construction, considering the low recall value of 0.10. It fails to identify 90% of the terms that (should) have been included in the query.

We conclude that the CRF model is able to predict *good queries* with a similar accuracy as *user-generated queries*. However, given the small distance to a trivial rejector for learning *good queries* and in particular

the low recall value, we do not consider the CRF model as a suitable approach to learn *good queries*.

4.3 PARAGRAPH EXTRACTION

The first step towards identifying the paragraph in focus is the extraction of all paragraphs in a Web page. This is similar to Web page segmentation [162], which aims to divide a Web page into its building blocks. These building blocks are coherent fragments of the page representing distinct information, such as navigational menus, advertisements or actual content blocks. While conceptually similar, Web page segmentation and paragraph extraction differ in the extracted blocks. Web page segmentation is about extracting coherent information blocks, whereas paragraph extraction concerns (longer) textual blocks. Navigational menus or images are not of interested for the extraction of paragraphs. Further, while Web page segmentation may extract the main content part of a page as a single block, for paragraph extraction the individual paragraphs of a (potentially large) content block matter.

4.3.1 *Related Work*

Web page segmentation and paragraph extraction in terms of automatic splitting of larger documents into smaller, semantically coherent parts, have been investigated before [30, 70, 191], but to the best of our knowledge not in the context of real-time processing of general Web pages. Comparisons of well-known Web page segmentation methods are available in the literature [103, 162]. Web page paragraph extraction is especially challenging for two reasons: First, markup tags are used to convey both, semantics and layout information[8]. Second, the storage capabilities and processing power within web browsers are limited, making real-time paragraph extraction based on content [70] or visual building blocks [30] intractable. BlockFusion [99] is most similar to our approach by using text as major source of information. It considers textual page elements as atomic blocks and fuses adjacent blocks with similar textual density. Lagun and Agichtein [107] suggested manually engineered segmentation (i. e. hard-coded) for popular Web pages which tend to share the same layout, such as Google search results, and supervised automatic classification based on selected training data for less frequent pages. However, our scenario of zero-effort retrieval on the Web requires the paragraph extraction to be generally applicable to all kinds of Web pages and feasible online.

8 Even though semantic elements (like article, section, ...) have been introduced with HTML5 (`https://www.w3.org/TR/html5` – last accessed March 2020), their use is neither enforced nor guaranteed.

4.3.2 *Approach*

Due to the limited computation resources within the browser, we rely on a heuristic, based on textual elements of the page's DOM (Document Object Model) tree. First, we collect potential candidates for paragraphs. A candidate is required to reside inside the body element of the page and needs to have a textual child node with a minimum length of 40 characters. Only the topmost node in a sub-tree satisfying these criteria is considered, since it already includes all child elements and therefore contains smaller potential paragraph candidates. For example:

```
<p>This sentence is longer than 40 characters. Accordingly, the
p-element is a candidate, while  <a>this link</a> is not.</p>
```

Second, paragraph candidates are combined and filtered. The whole process is detailed in algorithm 1 on the facing page.

Lines 3-7 define the collection of paragraph candidates. Besides the minimum requirement of 40 characters, we filter out elements, whose text is not displayed on the Web page, such as CSS definitions (<style>) for example. The actual candidate is not the text itself, but the parent DOM element. In the example above, the candidate paragraph is the <p> element and all its contained children. After a paragraph candidate has been found, we do not descend further down this sub-tree (described as post-filtering in lines 8-13 for readability, but achieved by a TreeWalker[9] that does not descend further down the tree in the actual implementation).

Lines 14-18 merge immediate neighbors, i.e., candidates that are not separated by any DOM element other than text.

We post-filter the set of candidate paragraphs. We keep all merged paragraphs (without further restrictions) and all non-merged that contain a dot and have a length of at least 100 characters (Lines 21-23). In addition, we filter for visibility (Lines 24 & 25). The most accurate way to determine whether a paragraph is visible, is to check whether it consumes space in the final layout of the page. A drawback of filtering for visibility is that this check requires (re-)rendering the element, which introduces additional computation costs. However, for paragraph candidates that are not visible it reduces the computational costs of collecting the actual paragraph contents and corresponding headline (see below) in turn. Still this check does not guarantee visibility, for example for white text on white background or text with full transparency. In our scenario, the user is in control to correct errors (cf. section 4.1.2.3 on page 32) and therefore we sacrifice potential errors regarding visibility over speed, omitting further checks.

We collect the corresponding (potential) headline for each extracted paragraph by traversing up the DOM tree and assigning the most

9 https://dom.spec.whatwg.org/#treewalker – last accessed March 2020

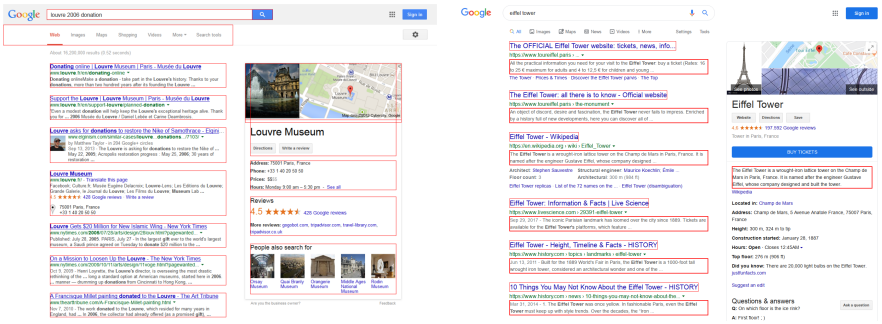
```
Data: DOM tree root element w (<body>)
Result: set of paragraphs P
1  begin
2      minChars ⟵ 40;
       /* candidate paragraph collection                          */
3      for node n ∈ DOM-Tree(w) & n.isTextNode() do
4          if n.parent() ∉{<style>,<script>,<noscript>}
5           if n.textLength() > minChars
6             P ⟵ P ∪ {n.parent()}
7      end
       /* keep only topmost (in tree) paragraph candidates        */
8      for p ∈ P do
9          for a ∈ p.Ancestors() do
10             if a ∈ P
11               P = P \ {p}
12         end
13     end
       /* candidate paragraph combination                         */
14     for p ∈ P do
15         while p.directNeighbour() ∈ P do
16             merge p and p.directNeighbour()
17         end
18     end
       /* candidate paragraph filtering                           */
19     minChars ⟵ 100;
20     for p ∈ P do
21         if not p.isMerged()
22          if p.textLength < minChars | not p.containsDot()
23            P = P \ {p}
24         if notVisible(p)
25           P = P \ {p}
26     end
27 end
```

Algorithm 1: Paragraph extraction algorithm

nearby <h> element to the paragraph. Finally, the DOM text nodes of the paragraph element and all its child elements are concatenated to represent the textual paragraph content.

4.3.3 *Results & Discussion*

Figure 4.4 on the next page illustrates extracted paragraphs for an exemplary Web page. Figure 4.4a on the following page highlights the page segments extracted by the approach of Lagun and Agichtein [107] by framing them with red boxes. In a similar fashion, we also frame the paragraphs and corresponding headlines extracted by our heuristic with red boxes (figure 4.4b on the next page). While our approach has found a headline "Description" corresponding to the box on the right ("The Eiffel Tower is a wrought-iron lattice..."), this headline is



(a) [107, figure 3] (b) Algorithm 1 on the previous page

Figure 4.4: Example for visual comparison of our heuristic (b) and the approach by Lagun and Agichtein [107] (a).

not visible in the rendered page layout. Since the query "louvre 2006 donation" did not result in a layout similar to figure 4.4a anymore (content on the right was missing), we adapted it to "eiffel tower". We further dropped search verticals such as images, videos or related news articles, in order to have a similar layout for comparison.

In our scenario, paragraph extraction (and subsequent focus paragraph detection) must be feasible online, i.e., immediately when the user visits and reads through a page. Hence, runtime is a crucial factor that we analyze for different types of pages and hardware in the following section 4.3.3.1. The other important factor is the quality of the extracted paragraphs, which we evaluate in section 4.3.3.2 on page 44.

4.3.3.1 *Runtime*

The runtime is mainly determined by the number of DOM (text) nodes, as those need to be evaluated as potential candidates and the number of extracted paragraphs, as for those the textual content and the corresponding headline need to be collected. Empirically, we found no difference in performance between a wide and a deep DOM tree in a worst case evaluation with a constant number of total nodes. In this worst case evaluation, all nodes have to be visited, whereas a deep tree can be beneficial if a potential candidate is found on an upper level of the tree and the algorithm does not need to descend down this subtree.

Web page auditing tools such as for example Lighthouse[10] recommend the DOM tree size not to exceed a total of 1500 nodes, a maximum depth of 32 nodes and a maximum of 60 children per node (width). For the runtime evaluation, we selected three prototypical Web pages: one that stays well within these limits, one that exceeds these limits, but can be considered a typical Web page and one that

10 https://developers.google.com/web/tools/lighthouse/audits/dom-size – last accessed March 2020

heavily exceeds these limits and additionally contains a lot of text. We refer to these prototypical example pages as simple, typical and heavy and detail them in the following (maximum values given for depth and width).

SIMPLE Landing page of the World Wide Web Consortium[11] as an example of an organizational overview page.
element nodes: 547, text nodes: 775, depth: 18, width: 29.

TYPICAL Google search result page[12] for the query "eiffel tower" as used in the example in figure 4.4b on the preceding page. Although this page exceeds the recommended limits, it is representative of a page users commonly visit.
element nodes: 6680, text nodes: 1362, depth: 39, width: 25.

HEAVY Wikipedia page about Donald Trump[13]. During the evaluation, this page was within the top 50 of longest Wikipedia articles[14]. In particular its amount of text poses a challenge to our algorithm.
element nodes: 19501, text nodes: 18148, depth: 30, width: 806.

All runs were executed in the Google Chrome browser in version 78.0.3904.87 (JavaScript engine V8, version 7.8.279.19). The evaluation was carried out on four different machines of different performance classes and years. All machines were equipped with Intel CPUs (specific model in parentheses). In detail, the machines comprised two laptops from 2006 (T2500) and 2011 (i7-2675QM) and two desktop machines from 2012 (i3-3220) and 2018 (i7-9700K).

Table 4.5 shows the runtime in ms for all machine/page combinations averaged over 10 executions of the paragraph extraction algorithm, both for the visibility filter activated and de-activated.

Table 4.5: Runtime of algorithm 1 on page 41 for different page types and machines of varying age (in ms, averaged over 10 script executions).

page type	**simple**		**typical**		**heavy**	
visibility filter	on	off	on	off	on	off
desktop 2018	1.1	0.7	1.8	2.7	100	103
desktop 2012	2.3	1.4	5.1	4.7	290	277.3
laptop 2011	2.7	1.5	5	4.6	226.4	193.2
laptop 2006	4.9	2.5	12.3	13	566.9	462.3

Nielsen defines a response time limit of 100 milliseconds "for having the user feel that the system is reacting instantaneously" and a limit

11 https://www.w3.org – last accessed March 2020
12 https://www.google.de/search?q=eiffel+tower – last accessed March 2020
13 https://en.wikipedia.org/wiki/Donald_Trump – last accessed March 2020
14 https://en.wikipedia.org/wiki/Special:LongPages – last accessed March 2020

of 1.0 second "for the user's flow of thought to stay uninterrupted, even though the user will notice the delay." [138]. The runtime of our paragraph extraction algorithm is well below the 100 ms limit for the simple and typical page type, even on a more than 10 years old laptop. For the page type "heavy", the runtime on a recent higher-end desktop machine coincides with the 100 ms limit. The older machines cross this limit, but stay below the 1.0 second limit.

The runtime without the visibility filter is mostly lower. Therefore, the filter may be turned off, sacrificing accuracy over runtime, if the interface allows the user to correct the paragraph identified as focused. Here we refer to *focused* paragraph instead of *extracted* paragraph, since a user does not notice a wrongly extracted invisible paragraph as long as it is not deemed focused.

For the above-mentioned response times, we also need to consider the initiating action. In our scenario of zero-effort queries, the user does not explicitly invoke the paragraph extraction. Instead, the extraction is triggered implicitly when the user visits a page. Therefore, the initiating action is the page visit, e.g. by typing the URL in the location bar. Hence, the runtime of the extraction algorithm adds to the total time required to load the page. On the other hand, contents of the page are already displayed to the user, even when the page is loaded only partial and the extraction is invoked only after the page is fully loaded. For example, the total time to load (and render) the heavy page on the 2011 laptop takes around 3 seconds, whereas the First Meaningful Paint (FMP) occurs at around 0.5 seconds. FMP refers to the page's primary content appearing above the fold (the area visible without scrolling) [161]. That means, the perceived response time in this example is 0.5 seconds and executing the paragraph extraction after the page is fully loaded does not alter the FMP and perceived response time respectively. The user will not notice delays caused by the extraction of paragraphs if either the runtime is small enough (below 100 ms) or the user does not interact with the page during extraction. Such interactions comprise for example scrolling. With the older machines, a "page freeze" may be noticeable on the heavy page type, if the user tries to scroll during paragraph extraction. However, besides the heavy page type representing a (rare) extreme case, the user may orient on the page first before interacting, giving the extraction algorithm time to complete.

In conclusion, the runtime of our extraction approach is small enough to not be noticed by the user for the simple and typical page type and only noticeable when interacting with the page for the heavy page type.

4.3.3.2 *Extraction Quality*

We evaluate the quality of extracted paragraphs with the collected dataset we described in the beginning of this chapter (cf. section 4.1.2.3

on page 32): Extracted paragraphs are framed with a dotted gray border and the paragraph deemed in focus is framed green. If users disagreed with the paragraph detected as focused (green), they could switch to another extracted paragraph (gray), by simply clicking. If none of the extracted paragraphs was suitable to the particular part of the text they were currently looking at, they were asked to adapt the extraction. This adaption was achieved by selecting the text, that should constitute a paragraph and mark it accordingly. We consider the paragraph extraction as correct, if a user did not change the detected focus paragraph or changed it to a paragraph extracted by our algorithm.

40 out of 251 paragraphs have been modified by users, i.e., 84% were extracted correctly. We manually investigated the 40 cases were users modified the extracted paragraphs. In one case, the user selected a single term only, in another case, a user selected the headline of a paragraph instead of the paragraph itself. Both cases are not covered by our approach, neither are they intended to be covered. In one case, our approach failed by not extracting a paragraph at all. The text passage selected by the user which should have been extracted as paragraph was a list of actors and their roles in a movie. In 6 cases, the text passage selected by the user was completely identical to a paragraph extracted by our approach. Obviously, these users did not understand that they could switch to another extracted paragraph with a simple click, but selected the text instead. Taking this observation into account, the percentage of correctly extracted paragraphs raises to 86%.

Five times, users extended the extracted paragraph. In one out of those five, our approach missed to include the first sentence in the extracted paragraph, whereas in the remaining 4, several sub-paragraphs that our approach identified as belonging to the next paragraph were added. On the contrary, the majority of modifications (26) comprised reducing the size of the paragraph. Mostly, a single sub-paragraph (a paragraph candidate that was merged with its neighbors - lines 14-18 in algorithm 1 on page 41) was selected as relevant. The only two exceptions were a single sentence missed by our approach and a user selecting several sub-paragraphs as relevant. Summarizing these observations, our approach failed in 3 cases, extracting at least meaningful paragraphs in 99%.

The analysis of extraction quality presented here has limitations that should be discussed. First, since the study was limited to Wikipedia pages, the results are only valid for pages with similar structure and (DOM) complexity. Pages with a huge amount of dynamic content are an example were our extraction approach would fail, as it is only executed once after the page is loaded. The frequent content change could be addressed by running the extraction algorithm repeatedly or by watching changes in the DOM tree. Second, we visually out-

lined extracted paragraphs and did not ask whether they are correct in general, but asked users to correct wrong paragraphs (requiring manual effort). Still, 29 out of 69 users adapted paragraphs, even with modifications such as adding only a single short sentence.

4.4 FOCUS PARAGRAPH DETECTION

Having extracted the paragraphs as described in the previous section 4.3 on page 39, we now need to determine the focused paragraph. In this section, we present a naive baseline and evaluate its accuracy based on the data collected in section 4.1 on page 29 as well as with an eye-tracking study. Based on the eye-tracking study data, we investigate further features to improve focus paragraph detection.

4.4.1 *Naive Baseline*

After paragraphs have been extracted by our heuristic as described in section 4.3.2 on page 40, we select the topmost (completely visible) paragraph as focus paragraph. This selected focus paragraph is updated only on scroll-events, i.e., when another extracted paragraph becomes the topmost paragraph in the page.

We used this baseline for several reasons: First, the topmost paragraph is the most likely paragraph to be looked at after following a hyperlink. Second, the topmost area of the page is already excluded by the paragraph detection algorithm if it merely contains navigation elements or advertisements. Third, updating the selected focus paragraph only on scroll-events guarantees the focus paragraph to not change accidentally. We therefore avoid switching back and forth between different paragraphs.

In the user study described in section 4.1 on page 29 we asked users to correct the focus paragraph if they considered it to be incorrect. Either by selecting a different paragraph extracted by our heuristic (cf. section 4.3.2 on page 40) or, if they did not deem any of the extracted paragraphs appropriate, by selecting an individual text passage. Obviously, if no appropriate paragraph has been extracted, it is impossible to correctly identify the focus paragraph. Accordingly, the evaluation of focus paragraph detection is limited to the 211 correctly extracted paragraphs (cf. section 4.3.3.2 on page 44). Users changed the focus paragraph in 73 of those 211 paragraphs, resulting in an accuracy of focus paragraph detection of 65%. Taking the incorrectly extracted paragraphs into account, the accuracy drops to 55%. This accuracy value is biased for the same reasons mentioned in the evaluation of extraction quality in section 4.3.3.2 on page 44: We did not ask whether the paragraph is correct in general, but asked users to correct wrong paragraphs (which took manual effort) and we visually outlined the detected paragraph, which might draw the focus of attention.

4.4.2 *Related Work*

Previous work on focus paragraph detection investigated the correlation between mouse interactions (movements, clicks, text selections) on a Web page and gaze with the goal of estimating the reading time spent on single paragraphs in the context of adaptive hypermedia systems. Hauger et al. [69] could predict the focus paragraph with approx. 79 % accuracy a-posteriori. Mouse clicks and text selections were identified as highly indicative for gaze position. Additionally, the authors found that the mean value of vertical eye position on web pages is not centered, but slightly shifted towards the top (mean value at pixel position 473 of 996).

The relationship between cursor and gaze has also been investigated for search engine result pages [76]. The authors approximated the x and y position of the gaze separately with a linear model, and achieved a RMSE of approx. 125 pixel in each direction. Their linear model combined the dwell time, the time since a movement, the x-coordinate of the cursor position, and the most likely x-coordinate of the gaze based on future cursor positions (analogous for the y-coordinate). Both authors (as well as [107]) use features that are not applicable for us, because they are not suitable for online processing (future mouse position) or for zero-effort querying (mouse click[15]). Further, a text selection already provides a very specific user context, rendering the paragraph detection unnecessary. While our application scenario is different, we conclude from related work, that i) mouse pointer is a semi-informative feature, ii) frequent mouse usage has to be identified and iii) vertical position seems to be highly indicative.

4.4.3 *Improving Focus Detection*

Although the first study showed an accuracy of the naive baseline algorithm of 65%, the study had several limitations as described before. In this section we describe our approach of improving the focus paragraph detection for the context of Web-based zero-effort queries using selected features based on related work and the results of an eye-tracking study. As mentioned in section 1.3 on page 7, the eye-tracking study is not the work of this thesis' author, but has been conducted by colleagues in Dresden. Accordingly, this section is joint work, to which this thesis' author contributed through paragraph extraction and (potentially) relevant features to improve focus detection.

15 A click typically corresponds to following a hyperlink, thus changing the page content. We regard clicks to the non-linked space as feature of user control, explicitly marking the paragraph as focused and hence not applicable to the automatic detection.

4.4.3.1 *Feature Selection*

We chose to investigate four layout and interaction features in detail (cf. figure 4.5): The **paragraph's size** $a = w \cdot h$ as a layout factor should account for the fact that larger areas on the screen have a higher probability of being looked at. The **paragraph's position** d (vertical and horizontal) on the page is included, because it has been identified as predictive for gaze on Web pages in a different scenario [69]. We also included the **position of the mouse pointer** relative to the paragraph m, which has shown to be of predictive value if users move their mouse frequently [76]. Thus, we also included a **mouse pointer activity** threshold, indicating whether the mouse is used as aid for cognition [69].

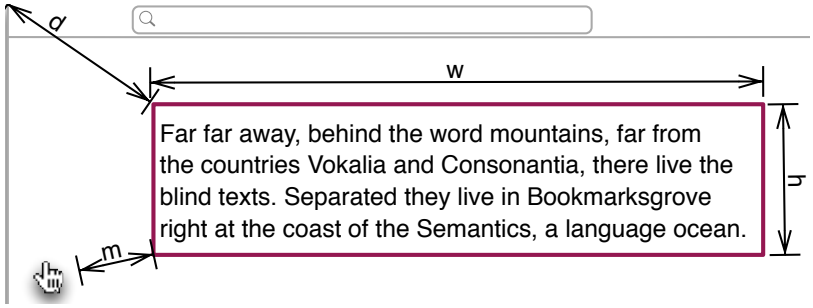


Figure 4.5: Features for focus paragraph detection.

4.4.3.2 *Eye-Tracking Study*

To investigate the applicability of the selected features (paragraph size and position, mouse pointer position and activity) we conducted a user study using an eye-tracker to log the visual focus of users while reading Web pages. The initial question was: which feature(s) could improve the accuracy of focus paragraph detection, and to what extend?

PROCEDURE The user study was conducted in a university lab using a Tobii Pro TX300 eye-tracker (providing a 23″ monitor). Twelve participants volunteered (2 female; age between 21 and 51; mostly students and staff from the local university), four needed a vision aid and used it during the experiment. Each session lasted around 30 minutes. First, the participants had to fill out a questionnaire about their web browsing and retrieval experiences and skills, and their knowledge and interest in 8 predefined topics such as geography, history, politics, etc. From these topics we later picked 2 or 3 for the session tasks (depending on the user's rating, preferring topics where the user indicated high interest and low or medium prior knowledge). The participants were given basic information about the setup of the study (accompanied by an individual calibration of the eye tracker),

but no further details about the purpose. So all subjects knew that their eye movements were tracked and logged, but did not know what exactly we were interested in. Afterwards, the participants were given three questions within the scope of the selected topic and a Wikipedia article[16] in a full-screen browser window on the Tobii monitor (full-HD). The article directly provided the answers to the questions or links to other relevant articles. The task was to read the relevant Wikipedia articles (without time limit) until one felt to be able to answer the questions. We were able to do at least 2 iterations (topics) with each subject.

We tracked and logged coordinates and time stamp of gaze and mouse pointer positions on the screen with 50 Hz, the URL and content of the visited Web page including position and layout of all extracted paragraphs and detected focus paragraphs (naive algorithm), as well as scrolling and mouse click activities (cf. figure 4.6 for some examples). The subjects were instructed not to use the browser's full

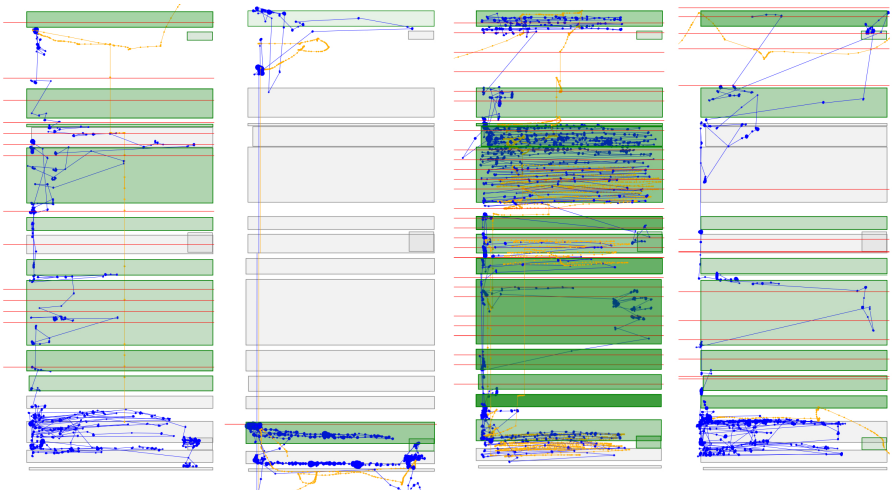


Figure 4.6: Four participants reading the same Web page: gaze hits (blue), assumed focus paragraphs (green), vertical scrolling position (red) and mouse activity (orange).

text search functionality to highlight keywords, so that they were forced to read larger parts of the text to find relevant information.

RESULTS In order to investigate the relevance and impact of layout and interaction features for focus paragraph detection, we analyzed the logged data as follows: First of all, we evaluated the performance of the naive focus paragraph detection approach by counting the number of gaze events which were targeted at a point within the bounding box of a focus paragraph candidate (extracted paragraph).

16 Please note that all participants shared the same native language, in which all reading material was provided.

To compensate inaccuracies of the eye-tracker[17] we extended the area of the bounding box by 10 pixels in each direction. Furthermore we smoothed the samples using a weighted average filter and removed short fixations (<100 ms), which are usually a result of noise but are especially dispensable in our case as we are interested in sequences of focused reading (at least 100 ms up to 500 ms fixation duration [179]). We learned that an average of 36.63 % (SD 12.93 %) of the gaze events targeted to the screen did not hit an extracted paragraph at all. This is due to the fact that elements like images, headlines, formulas, table of contents, etc. do not provide sufficient textual content to construct the search context for a query formulation, and are thus ignored by the algorithm (as described before). Taking this into account, we only considered gaze events within the set of extracted paragraphs. This resulted in an average success rate of 40.71 % (SD 18.27 %) for the naive baseline algorithm. We found that the lowest accuracy (9.5 %) was achieved when the subject was reading thoroughly through large parts of text without any mouse interaction or scrolling, which would trigger the reallocation of the focus. The best result (87.9 %) was achieved when the subject used anchor links within the document to directly jump to a paragraph and read it.

Next, we investigated the influence of the paragraph's size. We calculated the ratio between the size of the actual read paragraph (fixation duration of at least 3 seconds) and the size of the largest visible paragraph on the screen. The average ratio was 0.56 (SD = 0.35), which means that the mean size of a focused paragraph was about a half of the size of the largest visible paragraph. Only an average of 33 % of the gaze events hit the largest visible paragraph in each case.

After that we focused on the influence of the vertical position of the paragraph on the screen[18]. Regarding the overall vertical distribution of gaze hits on the screen we found that gaze is primarily directed to the upper half of the screen (mean value at 402 of 1080 pixels from the top). We used the frequency values (see figure 4.7 on the facing page) to weight the visible paragraphs according to their vertical position when rerunning the focus paragraph detection based on the logs. In this experiment we achieved an average success rate of 41.59 % (SD 16.65 %), which is only slightly better than the baseline algorithm. According to expectations, this approach performed better in cases of thoroughly reading of long text passages.

Furthermore, we investigated the relation between gaze events and simultaneous mouse pointer positions (as one of the feature candidates). As to be seen in figure 4.7 on the facing page the mouse pointer is mainly placed near the vertical center (mean value at 472 pixels of

17 According to Tobii the TX300 has an average accuracy of 0.4° and a precision of 0.14° [200].

18 As the majority of the Wikipedia articles used in this study appear in a single-column layout we did not have sufficient data to evaluate the influence of the horizontal position.

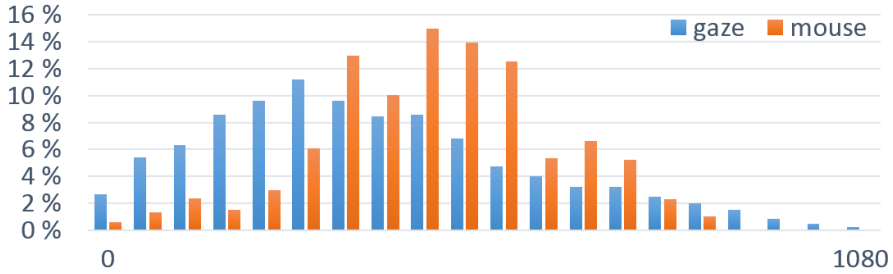


Figure 4.7: Histogram of the vertical distribution (0=top, 1080=bottom) of gaze (blue) and mouse pointer (red) positions

1080). We calculated a matching between mouse hovering in a vertical distance to the gaze target of under 100 pixels[19] in 38 % of all cases. According to the assumption that the activity of the mouse pointer is an indication for its usage as an aid for cognition [69], we focused on the samples showing high mouse pointer activity. In the selected subset we found a matching in 60 % of the cases.

4.4.3.3 *Discussion*

The results of the study are quite in line with our expectations, regarding the inferior performance of the naive baseline algorithm compared to the results of the first user study. This is mainly due to the limitations of the first study (cf. section 4.4.1 on page 46) and the difference in measuring the accuracy: while the first study achieved a binary measure through the (indirect) approval or (direct) rejection of the users without accounting for dwell time, the eye-tracking study measured gaze hits per paragraph in relation to time. The results confirm former studies [69, 76] concerning the deviation between gaze and mouse positioning, and that the mouse pointer is a semi-informative feature. The approach to use an activity threshold for the mouse pointer increases the accuracy significantly (up to 60 %) but was only applicable in 5 of 25 runs (20 %). The results showed that the relative size of a paragraph on its own is not an informative indicator for it being read. Furthermore, the weighting of the paragraphs according to their vertical positions based on the vertical distribution of gaze hits improved the accuracy only slightly, but seems to be promising in combination with navigational behavior of the users. This leads to the assumption that focus paragraph detection solely based on layout features is not feasible. Instead, future research should be focused on the weighted combination of layout, interaction and semantic features such as content-related background knowledge, browsing and query history, user profiles, etc.

The presented study has some limitations that should be further discussed. First of all, the number of subjects was too small to find similarities between users and derive adequate features in terms of the

19 The average height of a detected paragraph was 100 pixels.

analysis of typical reading behavior. Second, the subjects knew that their gaze was tracked and logged, which means that we cannot ensure that they acted normally, i. e., performed normal reading behavior, although all of them stated that they did not feel affected by the presence of an eye-tracker. Due to the complexity of the given tasks it is very likely that the participants forgot about the tracking after some minutes. On the other hand, this complexity and a certain pressure to succeed might be the reason that several subjects tended to skim the page to spot relevant keywords instead of attentive reading. We are not sure if this adds to a bias and to what extend.

4.4.4 *Conclusion*

The analysis of layout (paragraph size and position) and interaction features (mouse position and activity) revealed that those can increase the success rate of focus paragraph detection. However, based on layout features only, we saw only a small increase in accuracy from around 41% to around 42%. Taking mouse position into account, the accuracy could be increased up to 60%, but only for users with high mouse activity (20% of the runs in our study). Future research should focus on combinations of layout, interaction and semantic features (such as Web page content along with background knowledge, derived from browsing history or user profiles for example). For our scenario, we decided to keep the naive baseline, as its performance is only slightly worse compared to including further layout features. Furthermore, it has the advantage of transparency (selecting the topmost paragraph is comprehensibly to the user, whereas the selection based on paragraph position and size is less obvious) and leaves the user in control: If the detection fails, it can be corrected by the user and the focus paragraph selected by the user will not change until it leaves the viewport.

4.5 QUERY CONSTRUCTION

Having identified the paragraph in focus, we aim to find results relevant to that paragraph. More formally, we want to optimize the function $f = g \circ h : P \to R$, where P is the paragraph and R the result set, towards relevant results. The mapping from a paragraph P to a query Q is defined by $h : P \to Q$ and the retrieval of results by $g : Q \to R$. Less formal, the whole process is defined as $P \xrightarrow{h} Q \xrightarrow{g} R$. In just-in-time retrieval systems that treat the search engine as integral part, f is not a composition of g and h, but results are retrieved directly according to the paragraph ($f : P \to R$). In our work, we treat the search engine as black box and focus on the query side of retrieval. This means, we have no influence on g, but rather seek to optimize h

in terms of optimizing $f = g \circ h$. That is, we aim to construct a query that yields results relevant to the current focus paragraph.

During query construction, we need to keep in mind that search in digital libraries is fundamentally different from regular Web search. While Web search is typically a full-text search over indexed Web pages, search in digital libraries is typically a catalog-like search over (rich) metadata. Digital library search offers the ability to precisely filter and narrow down queries for the desired results, often specifically for particular domains. However this comes at the cost of low result quality if users cannot make use of those capabilities or are not aware of the relevant vocabulary, as natural language or simple keyword queries are highly likely to fail.

Some search engines aim to counter this effect by internally re-writing the queries and/or applying query translation. Figure 4.8 shows an example of such a re-written query in the PubMed database. The original keyword query "heart attack" is translated to the corre-

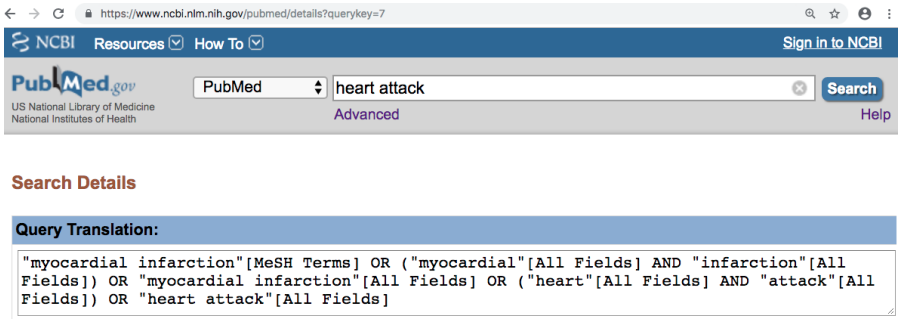


Figure 4.8: Example of query re-writing in the PubMed database. The original query terms "heart attack" are translated to the corresponding technical terms and the query is re-written as boolean query.

sponding technical term "myocardial infarction". Furthermore, the query is re-written as boolean query, performing a search with several term/keyword combinations across different metadata fields. "MeSH terms" refers to Medical Subject Headings, the controlled vocabulary thesaurus for indexing articles in PubMed. Besides searching for "myocardial infarction" in the MeSH metadata field, the boolean query searches for a combination of "myocardial" and "infarction" as well as for the phrase "myocardial infarction" in all fields. Similarly, the last part of the query repeats this for the original keywords "heart attack".

However, this support of automatic query translation/re-writing is not available by all digital library search systems and in particular smaller libraries cannot afford to maintain such a system. In order to address this challenge, we discuss different approaches and our solution by applying a particular query scheme in the following section 4.5.1 on the next page. We then present means to transform the context (focus paragraph) into a query conforming to this scheme in

section 4.5.2 on the facing page and evaluate and discuss the results in section 4.5.4 on page 58.

4.5.1 *Query Representation*

A paragraph P is represented by its sequence of words and a query Q by a set of keywords, which provide a compact representation of the paragraph. Furthermore, Q can be represented in two principled ways: either as a keyword query or as a boolean query. As described above, digital library search is typically not designed for keyword queries, posing a particular challenge to query construction. We distinguish between three major approaches for query construction and addressing this challenge:

1. **Keyword queries** are the simplest option to construct queries. They shift the responsibility to retrieve relevant results (and therefore potentially re-writing the query) to the search API provider.
2. **Query re-writing/translation** before submitting the query to the search API. Requires profound knowledge about the internal search mechanisms and contents of the particular provider.
3. **Boolean queries** allow to define the query more precisely than keyword queries.

The first option requires the search provider to have a query re-writing/translation system similar to the one illustrated by the PubMed example in figure 4.8 on the preceding page in place. In particular for niche content, this is rarely the case. The primary focus of small libraries, museums or archives is to curate their collection and provide access, not to maintain a sophisticated search system. Since we aim to support the discovery of resources in the long-tail of the Web, which are typically provided by those institutions, the first option is not viable. The second option is not viable either, as the relevant (internal) knowledge would require to implement a specific solution per provider, whereas we aim for a search engine agnostic solution. Therefore, we opt for the boolean representation, as it provides richer expressiveness (compared to keyword queries) and most digital libraries, which expose their contents via a search API, support boolean queries. In addition, boolean queries can be easily transformed into keyword queries (in case a provider does not support boolean queries), while the opposite is not possible in general. Combining all keywords with either OR or AND yields under- or over-specified queries, in particular, when the set of keywords grows. While the problem of over-specified queries is obvious (no results), the problem of disjunctive queries is less obvious: results, which are triggered by a single keyword only, may not fit the topic of the paragraph very well and hence

are quite unrelated. Moreover, results which are triggered by a single keyword can suppress results that are related to several keywords. The conjunctive normal form (CNF) provides means to formulate highly precise queries, which are not over-specified though, and hence yield results. However, finding the optimal query for arbitrary combinations of keywords in CNF is NP-hard and hence intractable in general. Therefore, we propose to formulate a boolean query in CNF of the following structure:

("main topic") AND ("keyword 1" OR "keyword 2" OR ...)

where the main topic is defined as the overall topic of the paragraph and the right part of the conjunction are additional keywords. This way, we can be sure, that a keyword triggers only results which are connected to the overall topic of the paragraph. Even though, from the perspective of the search engine, all of the query terms are keywords, we will refer to the left part of the conjunction as *main topic* and to the right part as *keywords* in the further course.

4.5.2 *Approach*

Having defined the query scheme in the previous section, we now describe how both parts, i.e., the main topic and the keywords, can be extracted from the focus paragraph and how the resulting query can be optimized.

4.5.2.1 *Extraction of Keywords*

Keyword extraction algorithms that represent the keywords in terms of a subset of terms from the original text are available in the literature [131, 158]. However, query log analysis research revealed, that over 71% of (user generated) search queries contain named entities [64] and named entities exhibited good performance in related work [109]. In addition, named entities have been show to be beneficial to query segmentation [67], a technique that is used to optimize queries. Also, named entity extraction can be seen as some kind of keyword extraction task, as the original text is represented by a smaller set of terms. Therefore, we base our query generation on named entities, which are obtained via DBpedia Spotlight[20].

4.5.2.2 *Extraction of Main Topic*

For the extraction of the main topic, we utilize Paragraph Vectors [108], also known as Doc2Vec. We will describe the models of Doc2Vec in detail in section 10.2 on page 144 and section 13.1 on page 175. For the discussion here, it is sufficient to know that Doc2Vec provides a feature vector representation for a piece of text. Based on Word2Vec [133],

20 http://spotlight.dbpedia.org/ – last accessed March 2020

Doc2Vec produces an embedding vector, given a sentence or document. Hence, we use the entire input paragraph and infer a vector representation given a Doc2Vec model created on a Wikipedia corpus. We compare this vector with the Doc2Vec representations of all named entities extracted from the paragraph via DBpedia spotlight (i.e. also Wikipedia pages) by computing the cosine similarity. We restrict the similarity computation to the extracted named entities instead of all possible entities contained in Wikipedia for performance reasons. The named entity with the highest similarity to the input paragraph represents the main topic. The remaining named entities are used as keywords in the right part of the boolean conjunctive query. The named entity extraction and similarity calculation is provided by the DoSeR framework [229, 230].

4.5.2.3 *Selection of Keywords*

Having extracted the keywords and main topic, the baseline approach to a query in the defined CNF is to use all of the extracted named entities as keywords, to which we will refer as MTAK (main topic and all keywords) in the further course. However, this may still yield some irrelevant results, not least as the named entity extraction is not perfectly accurate. The optimal query (MTBK, main topic and best keywords) can be obtained with a brute-force approach. Knowing the optimal query, we know the optimal selection of keywords to use in this query. Based on this knowledge, we can train a classifier, that predicts whether a keyword should be used in the query or not. We refer to those queries as main topic and predicted keywords (MTPK).

4.5.3 *Own Search Index*

In our experimental setting, we stored the results retrieved during the dataset collection (cf. section 4.1 on page 29) in our own search index for several reasons: First, the collection of a repository is subject to change (new items may be added or the ranking may change), which would render future comparability of the results infeasible. Second, to identify the optimal query for a set of keywords (MTBK as defined in the previous section 4.5.2.3), we need to issue an enormous amount of queries and collect corresponding results. Last, as we set up our index with results retrieved during the data collection, we have relevance ratings available for all results in the index.

We created a separate Elasticsearch[21] index for each of the two providers (Mendeley and Europeana - cf. section 4.1.2.5 on page 34) in the dataset collection. For Europeana, we crawled the full metadata for every record retrieved during the study and used this data to populate the index. For six records, the data could not be obtained – they seem

21 https://www.elastic.co/elasticsearch/ – last accessed March 2020

to have been removed from Europeana in between the user study and the crawling. Mendeley already returned the full content-based metadata during the search in the user study. Hence, we used this data to populate the index. To answer queries, at most the top-10 results were obtained from each index and interleaved via Round-Robin just as with the original index (cf. section 4.1.2.5 on page 34).

4.5.3.1 *Identifying Optimal Queries*

For all paragraphs in the collected data, we identified the maximum achievable performance of the proposed approach. That is, we collected the optimal queries (MTBK) that can be posed based on the extracted main topic and keywords and the query scheme as defined in the previous section 4.5.1 on page 54 and section 4.5.2 on page 55. If the set of candidate keywords was sufficiently small, we performed queries against our own search index with all possible combinations of keywords. Otherwise we selected promising candidates, according to the strategy described in the subsequent paragraph and evaluated all their combinations. The combination of candidate keywords which performed best in terms of F1-score was chosen as best query.

CANDIDATE SELECTION The approach for selecting good keyword candidates is as follows: First, we select all keywords, that yield positive results with a boolean AND query combined with the main topic. That is, we transform the boolean query ("main topic") AND ("keyword 1" OR "keyword 2" OR ...) from section 4.5.1 on page 54 to the equivalent form ("main topic" AND "keyword 1") OR ("main topic" AND "keyword 2") OR ...". If none of these queries yields positive results, we try again with the keywords only. In the second step, if the candidate set is still too large, we remove those candidates, that do not yield additional positive results, compared to the results retrieved by all other candidates. Finally, if the candidate set is still too large, we restrict it to the candidates that retrieve results from both providers. The threshold for the maximum candidate size in our case are 20 candidates, since this is the maximum amount of results returned (cf. section 4.1.2.5 on page 34, top-10 results per provider). Hence, with 20 candidates, every candidate can at most contribute one unique item and contributions of further candidates would not show up in the final result list. At the same time, this is the justification for our second step (removing candidates that do not yield additional results). Still, the combinations to be tested amount to $\sum_{k=0}^{n} \binom{n}{k}$, where n is the number of candidates and k is the size of the combination set to be tested. This equation is equal to 2^n and over one million combinations to test for a set of 20 candidates.

4.5.4 *Results and Discussion*

The evaluation comprises a comparison of our own search index with the original search engines and the suitability of the extracted main topic. Further, we evaluate the query quality of the baseline approach for automatic queries with all extracted keywords (MTAK), the advanced approach with predicted keywords (MTPK), the optimal queries (MTBK) and the best queries, users were able to formulate (USER*). In the evaluation, we treat the relevance feedback provided by the users as a binary measure: results rated with *not relevant* and *cannot judge* are considered as not relevant, results rated with *relevant* and *perfect match* are considered as relevant.

4.5.4.1 *Comparison of Own Search Index and Original Repository*

Since the collection of our search index is only a subset of the collections of the original repositories and we are not able to fully reflect the original ranking, the result sets retrieved via our own index are not perfectly identical to the original ones. In order to get insights into the differences between the original and our own index, we compared the results retrieved from both on a per page level. Per page level means, that we compare the union of all result sets retrieved by all queries performed against an index on a particular page. A result is considered as true positive, if it is contained in both union sets, false positive, if it is only contained in the union set of our index and false negative if it is only contained in the union set of the original index. The comparison is performed separately for each provider. Table 4.6

Table 4.6: Difference between Own Search Index and Original Repository.

	Mendeley	**Europeana**
precision	0.82	0.98
recall	0.55	0.85
recall of positively rated	0.64	0.82

shows the micro-averaged precision and recall values of the comparison. Also, the micro-averaged recall values of results rated as relevant are indicated. We report the recall values of positive results, as we are in particular interested in those, since it is important to retrieve relevant results, while missing irrelevant results can be ignored or even seen as beneficial. As can be seen in table 4.6, the values for Europeana are higher than those for Mendeley. We account this mostly to the querying behavior: while we use boolean queries for Europeana, the query to Mendeley is a keyword list (cf. section 4.1.2.5 on page 34). Hence the difference in the collection shows a larger effect for Mendeley, as it has a larger impact on the ranking. On average, the precision

amounts to 0.90, the recall to 0.70 and the recall of positive results to 0.73.

4.5.4.2 *Suitability of Main Topic*

In order to determine, how well the extracted main topic is suited as main topic in the query, we evaluate, how often it has been changed by the users in the study. Changes occurred in 67 out of 251 automatic queries. This indicates, that users agreed with the choice of main topic in 73% of the queries.

However, 107 queries have not been modified at all. 32 of these 107 were followed by a change of the paragraph, resulting in a new automatic query. The remaining 75 queries were the sole queries within a task. For 14 out of those 75, users indicated, that they were perfectly satisfied with the results, and for 14, they indicated that they do not deem the search engine to be able to deliver any results. 34 queries have not been modified due to a timeout and for 13 queries, no reason was provided. If we leave out the queries, where we cannot make a statement about the main topic quality (i.e., queries not modified due to timing constraints or no reason provided), the fit of the main topic drops to 67%.

On the other hand, a change of the main topic led to an immediate improvement of the results (in terms of average rating) in only 42 cases. If we account for this finding, the suggested main topic is appropriate for 83% of the queries (79% without timeouts and no reason provided).

As the main topic is represented by a Wikipedia article title and the evaluation was carried out on Wikipedia pages, we can easily compare the main topic extracted from the paragraph with the topic of the page. In 103 queries, both were the same. Moreover, the suggested main topic was different from the page topic in 54 of the 67 queries where the main topic was changed and the change resulted in an improvement for 34 queries. Also, the extracted keyword with the page topic was set as new main topic for 17 queries.

These findings suggest, that in general, the topic of the whole page is well suited to be used as main topic in the query. Therefore, under the assumption that the main topic is extracted correctly, main topic extraction should be based on the whole page rather than the focused paragraph.

4.5.4.3 *Classifier for Keyword Prediction*

In the previous section, we described a potential improvement for the first part of the query, the main topic. For the second part of the query, the keywords, two modifications are possible: addition or removal (modification of a keyword is seen as remove & add). Removal is the predominant modification, users applied to queries in the dataset, which justifies the approach of predicting, which keywords

should be used in the query (MTPK) as the prediction reduces the set of keywords. In 111 cases, both modifications were applied. The frequency of removing keywords only (119) is almost three times the frequency of adding keywords only (42). Also, nearly twice as many keywords have been removed (1524) than added (843).

We trained a decision tree classifier on the set of optimal queries (MTBK), in order to predict the relevant keywords (MTPK). The features we used for classification comprised the frequency of the keyword, its similarity to the main topic and other keywords and its position(s) in the paragraph. The frequency is the amount of occurrences of the keyword in the paragraph. The similarity to the main topic is defined by the cosine similarity between the Doc2Vec representation of the keyword and the Doc2Vec representation of the main topic. Similarly, the similarity to other keywords is defined by the cosine similarity of the Doc2Vec representation of the keyword and an average Doc2Vec vector of the remaining keywords, including the main topic. The position(s) in the paragraph determine(s) where in the paragraph the keyword occurs, normalized by the length (in terms of characters) of the paragraph. We cross-validated the classifier on a 10-fold random split over the original/best query pairs, yielding an average F1-score of 0.71. We present the evaluation of the resulting queries (MTPK) in the next section.

4.5.4.4 *Query Quality*

We evaluated the query quality of the baseline approach for automatic queries with all extracted keywords (MTAK) and the best queries, users were able to formulate (USER*) on the original data. We further evaluated these two (MTAK and USER*) on our own index together with the best achievable queries (MTBK) and predicted keywords (MTPK). In all evaluations, we report precision-, recall- and F1-scores macro-averaged over all queries (hence the F1-score is below the value obtained from the average precision- and recall-score).

Clearly, we cannot measure the true recall value, as we do not have ground truth relevance feedback for the whole collection. Instead, we approximate the recall with all positive results retrieved via all queries executed in the context of a particular page. Similarly, not all items have been rated by every user in the context of each paragraph. We synthesize missing rating values with the ratings from all users on a page. When only a single rating exists for the item on a page, or all ratings are equal, we set the item's score to this value. Otherwise, we remove all *cannot judge* ratings and check again for the just mentioned condition. By removing *cannot judge* ratings, we favor the opinion of users that are able to judge the relevance. In case the condition is still not satisfied, we take the value with the largest agreement. If the value with the largest agreement is not unique, we take the mean.

ORIGINAL INDEX The results on the original index for automatic queries (MTAK) and the best queries users were able to formulate (USER*) are presented in table 4.7. For the evaluation of user queries,

Table 4.7: Comparison of MTAK and USER* on the original index.

	MTAK	**USER***
precision	0.29	0.33
recall	0.25	0.33
F1-score	0.24	0.31

we took the best query, a user was able to formulate for a paragraph (USER*). This means, that if the initial automatic query scored better in terms of F1-score than all subsequent modifications by the user, we take the initial query. Consequently, the automatic queries provide a lower bound for the best queries users were able to formulate. As can be seen from the table, the performance of baseline automatic (MTAK) and user queries (USER*) is quite low, with the user generated queries performing slightly better (0.31) than the automatic queries (0.24). In particular, if we consider, that MTAK is restricted to the extracted main topic and keywords, while users can provide arbitrary values for these two.

OWN INDEX Within our own index, items that have not been rated in the context of a page might be returned. As we cannot make a definitive statement about their relevance, we treat them as not relevant, even though they actually might be relevant. The true performance of the queries performed on our own index might be higher than reported. Following a pessimistic approach, the reported values provide a lower-bound instead. The results on our own index for automatic queries (MTAK), best user queries (USER*), best achievable queries (MTBK) and predicted keywords (MTPK) are shown in table 4.8.

Table 4.8: Comparison of MTBK, MTAK, MTPK and USER* on our own index.

	MTBK	**MTAK**	**MTPK**	**USER***
precision	0.55	0.34	0.40	0.37
recall	0.49	0.33	0.33	0.37
F1-score	0.49	0.31	0.33	0.35

As expected, the performance on our own index is higher than on the original index (cf. table 4.7), as the amount of potential false positives is reduced. Remarkably, the performance of best user queries (USER*) is far away from the performance that could be achieved based on

the best combination (and filtering) of extracted keywords (MTBK). While MTBK is limited to the extracted keywords, user were free to choose arbitrary terms. Still they are not able to even match the performance of the best combination of those predefined extracted keywords. While the F1-score of the keyword prediction approach (MTPK) is lower (0.33) than for the best user queries (USER* - 0.35), MTPK performs better in terms of precision (0.40 vs. 0.37). In zero-effort queries, we favor precision over recall, since we aim to present only relevant results to the user and not bother her with irrelevant results. Again, it is to note that MTPK is restricted to the extracted main topic and keywords, while a user can provide arbitrary values.

4.5.5 *Summary and Conclusion*

In this section, we presented our approach to transform the focus paragraph into a query to a search engine. To this end, we defined a boolean query scheme consisting of main topic and additional keywords that allows query re-writing for arbitrary search APIs, as long as they support boolean queries. The evaluation revealed that the main topic in this query scheme is to be extracted from the whole page instead of the paragraph in focus. We showed that automatic queries can be generated, which perform better in terms of precision than the best queries, users were able to formulate. The small improvement in terms of result quality of user-adapted queries at best and their performance staying below what is even possible with the set of extracted keywords once more shows the need to support users in their search for additional related resources.

4.6 PERSONALIZATION

Our approach to personalization is to subdivide large paragraphs into smaller sub-paragraphs and construct separate queries for each individual sub-paragraph. The (sub-)query with the highest overlap with the user profile is then sent to the search engine for resource retrieval. The reasoning for this personalization approach is that in particular large paragraphs lead to a large amount of extracted keywords (named entities). In order to filter this large set of keywords, we select the sub-set of keywords that is most relevant to the user (i.e., has the highest overlap with her user profile).

The sub-paragraphs are readily available by our paragraph extraction approach: When merging neighboring paragraphs, we simply store the merged neighbors as sub-paragraphs (cf. section 4.3.2 on page 40).

The user profile is based on the query history of automatic queries. We only take those queries into account, where the user viewed the retrieved results, as this indicates interest. To construct the profile, we

use the query's keywords, derived by extracting named entities from the paragraph via DBpedia Spotlight (cf. section 4.5.2.1 on page 55). These extracted named entities correspond to Wikipedia articles. Each Wikipedia article is assigned to one or more categories (the category assignment is shown at the bottom of a Wikipedia article). For example the article about the German football player "Thomas Müller" is (among others) assigned to "German footballers". This category assignment provides a layer of abstraction and we store the categories of all query keywords in the user profile, if the results of this query have been viewed. As in section 4.5.2.2 on page 55, the categories are obtained via the DoSeR framework [229, 230]. The overlap between the user profile and a sub-paragraph is then calculated by simply counting the matching categories between the user profile and the categories assigned to the query keywords of this sub-paragraph.

We took this approach as a sub-paragraph usually covers a particular aspect of the topic of the whole paragraph and by the highest overlap with the user profile, we deem this aspect to be the most interesting to the user.

4.6.1 *Results and Discussion*

Table 4.9 shows the average rating per task, both for the personalization and the non-personalization group. To calculate the score, results rated as "not relevant" are represented by -1, "cannot judge" by 0 and both, "relevant" and "perfect match" by 1, same as in section 4.1.3 on page 35. As mentioned in that section, users in the personalization

Table 4.9: Comparison between personalization and non-personalization group by average rating per task. Possible values: -1 (not relevant), 0 (cannot judge), 1 (relevant).

Task	personalization	no personalization
1	-0.23	-0.21
2	-0.26	-0.07
3	-0.16	-0.31
4	-0.22	-0.49

group tend to rate more positively in general (-0.16) than users in the non-personalization group (-0.23). Still, users rate more results as "not relevant" than "relevant". Those values are averaged over all queries, whereas the average values reported in table 4.9 only account for automatic queries.

4.6.1.1 *Comparison of (Non-)Personalization on Similar Content*

The most meaningful row for comparison is the last row (Task 4) for two reasons: First, since this was the last task, personalization has the highest effect on the queries. As personalization is based on the overlap of a query with the user profile and the user profile is constructed from past queries, no personalization can take place in the first task (since there are no past queries available and accordingly, the user profile is empty). Second, while in the first three tasks users were free to choose a page from a predefined set, the page was predefined and the same for all in task 4. That is, in task 4, queries were generated from the same input paragraph for both groups. It is to note, that the input paragraph is not necessarily exactly identical, as users were asked to navigate to a particular section of the page, but could still adapt the focus paragraph. The average rating score is higher in the personalization group and the gap is far more pronounced than the average over all queries (-0.16 vs. -0.23). Further, there is a tendency towards a better score as the tasks progress, while for the non-personalization group the opposite is the case. Therefore, we conclude that personalization is beneficial in terms of result quality.

4.6.1.2 *Impact of Search Intention*

Surprisingly, the average rating in the non-personalization group in task 2 is exceptionally high (-0.07). Our first interpretation was that this high value could be caused by different pages between the user groups (participants were free to chose from a set of predefined pages in the first three task). The reasoning of this interpretation is that a page for which highly relevant results are retrieved, but which is only present in one group might skew the score. However, limiting the average score calculation to pages that were present in both tasks and normalizing by their frequency resulted in similar values as presented in the table. In particular, the exceptionally high value in task 2 of the non-personalization group was still present.

We then manually inspected the pages and result ratings of both groups in task 2 and learned that majorly two pages were responsible for the high value. Dropping these two would result in scores that are closer to the rest of the table (-0.24 instead of -0.07 for task 2 of the non-personalization group). For these two pages, the majority of the personalization group rated negatively, whereas the majority in the non-personalization group rated positively.

The difference is grounded in the users' search intention (we asked user two provide their search intention in the accompanying questionnaire, cf. section 4.1.2 on page 31). Even though the focus paragraph was similar or the same for both groups, their search intentions diverged. For example, one of the two pages was about "Atheism" and the non-personalization group mainly described their search intention

as "looking for a general definition". On the contrary, search intentions in the personalization group comprised more detailed information, e.g. about different movements in atheism. This observation further highlights the need for personalization.

4.6.2 *Conclusion*

The results show that personalization has a positive effect on the result quality. Different search intentions on the same or similar content should retrieve different results as indicated by the divergent rating scores in the last section. Both observations highlight the need for personalization, whereas the second observation about search intention points to a further issue: The personalization group rated highly negative in the two pages investigated in terms of search intention, even though they had the personalization feature. This could be caused by two reasons. One is the cold start problem, i.e., the interests collected in the user profile might not yet have been sufficient to personalize adequately. The other is, that our user profile as defined in the beginning of this section 4.6 on page 62 might not be able to collect sufficient statistics about the user's interests. The user profile is collected from previous queries where the user viewed results. Hence it can only capture a partial picture of the user that is limited to exactly those queries, missing further user interactions on the Web. More importantly, it cannot capture any interests of the user for which she does not visit corresponding Web pages.

In order to bootstrap the profile for personalization and potentially capture user interests which are not observable by Web browsing interactions, we propose to populate the user profile with information obtained from online social media. In the next chapter, we present our approach for creating such a profile, even for passive users. That is, users that do not post own content on social media, but passively consume the available content.

5

PERSONALIZATION VIA USER PROFILES

The need for personalization as highlighted in the last section is a general phenomenon as we have seen a rapid increase in the amount of published information and data since the rise of the Internet. Obviously, it is not possible for humans to process all the information available, a problem known as "information overload" [49]. At the same time more and more people reveal their interests explicitly in and implicitly by using social networks. The goal of social media based recommendation systems is to infer users' interests and preferences from their social network activity and use the thereby generated interest profiles for making personalized content recommendations. Using social information for recommendation systems is also connected to the hope of solving the cold start problem which in particular correlation based approaches suffer from, especially for smaller Web pages. The cold start problem concerns the issue that a system does not know anything about new users and needs an initial phase to gather information about them.

Most of the related work infers the interest profiles from a user's posts or tweets. However, there might be a significant difference between what a user *produces* and what she *consumes*. Moreover the passive use of social network sites is on the rise. Now four in ten users browse Facebook only passively, without posting anything [63]. For those users, profile construction based on a user's postings fails, since there is simply no input from which the profile could be created. We address this problem by inferring semantic interest profiles from the Twitter followees (the accounts, the user follows) rather than her tweets. It is to note, that while we focus on Twitter and followees, the approach could be adapted to other online social networks as well, by accounting for the corresponding features, e.g. *likes* on Facebook.

The rationale for the followee-based approach is that many famous people maintain a Twitter account and a lot of Twitter users follow these accounts. For those accounts, the likelihood that a Wikipedia article about this person exists is very high. Moreover, Wikipedia articles are typically linked to higher level categories (e.g., the article about the football player "Thomas Müller" is linked to the category "German footballers"). Making use of those categories, following an account that can be linked to a Wikipedia article can be seen as implicit expression of interests (e.g., following the football player "Thomas Müller" reveals interest in "German footballers"). In addition, the assigned categories are organized in some kind of hierarchy in Wikipedia, thus they can be traversed in order to provide a more

fine- or coarse-grained profile. This approach immediately raises the question of whether a sufficient number of followees can be linked to Wikipedia entities, which we address in the first part of this chapter.

Specifically, this chapter is organized as follows:

1. We present an overview of our approach to (passive) user profile creation.
2. We evaluate the coverage of followee lists in terms of named entities in the English Wikipedia and show that the followee lists provide enough input to infer comprehensive semantic interest profiles.
3. We evaluate the quality of the created profiles, showing that those can compete with state of the art tweet-based approaches.
4. We compare the similarity of followee- and tweet-based profiles and show that they are more similar on very concrete and abstract levels than in between.

5.1 RELATED WORK

Research on user profiling and personalized content recommendation has been done for many years since the beginning of the Web [121]. Early approaches focused on the Web [112, 121] and search history [196] of the user. Recently, with the emergence of social networks like Twitter, research has shifted to analyze user activities on these platforms. For instance Siehndel and Kawase [185] introduced *TwikiMe*, a prototype for generating user profiles by extracting entities from the user's tweets and linking them to the 23 top-level categories of the English Wikipedia. This leads to abstract interest profiles with a fixed size represented as a 23-length vector.

Abel et. al. [1] in their work compared hashtag-based, topic-based (bag-of-words) and entity-based user models generated from the user's tweets, for news recommendation. In this approach the scoring of the extracted concepts and interests is based on a simple term frequency technique. The results of their comparative evaluation showed that the simple bag-of-words and hashtag-based approaches, which did not consider the semantics of a tweet, were clearly outperformed by the (semantic) entity-based strategy (precision of 0.71 compared to 0.4 and 0.1). Based on these results Tao et. al. [199] presented *TUMS*, a Twitter-based User Modeling Service, that tries to infer semantic user profiles from the messages people post on Twitter. However, the focus of *TUMS* is to make use of semantic web technologies for providing a standardized representation of the interest profiles allowing an easy exchange between different web services. This is connected with the hope to solve the so-called ramp up or cold start problem, a downside of approaches like content based or collaborative filtering [124,

199], which usually depend on the build-up of a user history before making personalized content recommendations. In terms of the applied algorithm and the knowledge base, the approach introduced by Kapnipathi et. al [91] is the closest to our work. They used the English Wikipedia to spot entities in tweets and leveraged the hierarchical relationships by performing a spreading activation on the Wikipedia Category Graph to infer user interests. The result, a weighted hierarchical interest profile (expressed as a so-called *Hierarchical Interest Graph*), was evaluated by a user study which showed an average of approximately eight out of the ten interests in the graph being relevant to a user.

Even though Siehndel and Kawase [185] suggested investigating other types of inputs for inferring user interests, most of the related work only makes use of the content posted by a user (e.g. the tweets). Some approaches tried to consider the social graph of the user at least to some extent [124, 148] whereas Lim and Datta [122] presented a basic approach for interest profile creation based on celebrities, a user follows. These celebrities are classified as belonging to one or more of 15 predefined interest categories. The classification is based on the celebrity's *occupation* field on his or her Wikipedia page and a set of keywords associated with each interest category. While this approach is also based on followees, in contrast to our work, it ignores the category information provided by Wikipedia and provides only support for a fixed (and predefined) set of interests, similar to Siehndel and Kawase [185]

Most similar to our work, a recent approach by Faralli et al. [51] also utilizes followees and the Wikipedia Bitaxonomy. However, while there are similarities in the applied approach, Faralli et al. do not directly evaluate the semantic interest profiles. Instead, they use it to identify users as belonging to a target population or not. Further, they apply itemset mining and based on the itemsets and association rules, they provide recommendations, e.g. for topical friends or categories a user might also be interested in and evaluate those recommendations. We in contrast evaluate, whether the constructed profile really describes the user.

5.2 APPROACH OVERVIEW

Our approach to interest profile creation can be seen as a four-step process which is shown in figure 5.1 on the next page. In the following, each step is described in more detail and a fictional user called *@soccerfan* will be used as an illustrating example.

FETCH USER'S FRIENDS In the first step the accounts which are followed by the user (the followees) are crawled. This is done

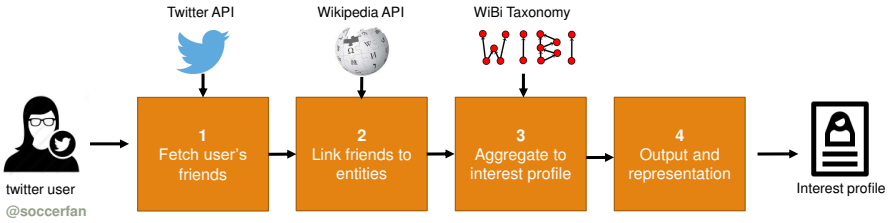


Figure 5.1: Interest profile creation overview.

through Twitter's RESTful Web API[1]. As the API applies strict rate limits, extensive use of caching techniques is made to reduce the number of requests sent to Twitter.

The fictional user *@soccerfan* might, among others, follow the accounts *@Cristiano (Cristiano Ronaldo)*, *@BSchweinsteiger (Basti Schweinsteiger)*, *@neymarjr (Neymar Jr)*, *@FIFAcom (FIFA.com)* and *@esmuellert_ (Thomas Müller)*.

LINK FRIENDS TO ENTITIES The objective of this step is to link the user's followees to corresponding entities represented by articles in Wikipedia. This entity linking includes handling coincidental homonymy and ambiguity (for instance there are several famous "Thomas Müllers" with their own Wikipedia page). For that purpose the MediaWiki Web API[2] is used and several disambiguation heuristics are applied. They include syntactical measures (overlap coefficient of last 20 tweets and article summary) and probabilistic heuristics (Sense Prior and a reverse linking of Wikipedia articles to Twitter search results).

In our example the following entities might be extracted: *WikipediaPage:Christiano Ronaldo*, *WikipediaPage:Bastian Schweinsteiger* and *WikipediaPage:Thomas Mueller (footballer)*. As you can see, "Thomas Müller" was correctly linked to the famous football player.

AGGREGATE TO INTEREST PROFILE The extracted Wikipedia article entities are assigned to Wikipedia categories. These categories are hierarchically structured (at least to some extent) and used to represent particular interests of the user. By performing a spreading activation algorithm on the Wikipedia Bitaxonomy (a taxonomy based on the Wikipedia page and category hierarchy [54]) the single interest entities are aggregated to a more abstract and broader interest profile.

The categories of the Wikipedia page entities extracted in the previous step represent the set of initially activated nodes. Their activation is spread during several iterations to neighboring

1 https://dev.twitter.com/rest/public – last accessed March 2020
2 https://www.mediawiki.org/wiki/API – last accessed March 2020

nodes connected by outgoing edges. Formally the activation $a(v)$ of a node v can be written as:

$$a_t(j) \leftarrow a_{t-1}(j) + d \cdot a_{t-1}(i) \tag{5.1}$$

where j is being activated by node i and $0 < d < 1$ represents the decay factor. If a node is activated by more than one node the activation is accumulated in this node. Apart from that, a normalization with the number of incoming edges and a so-called Intersection Boost (see [91] for more details), boosting nodes that are intersections of different paths are applied.

In our example the entities (pages) are assigned to categories such as *2014 FIFA World Cup players* or *German footballers*. Performing spreading activation identifies *sports* and *footballers* as two of the most suitable overall interest categories for the example user.

OUTPUT AND REPRESENTATION As the output of step three is a graph data structure with weighted nodes, the objective of this last step is to convert this representation to a common exchange format. Therefore, the top-k interests are extracted and can be represented in an arbitrary format. Typical representations include JSON or XML and semantic web vocabularies, such as the *FOAF*[3] (Friend of a Friend) or *Weighted Interests Vocabulary*[4] could be used. This also allows the provision of the interest profiles to other applications and web services through standardized interfaces.

5.3 ENTITY COVERAGE EVALUATION

The first question we need to address is whether the followee list of a Twitter user is sufficient input for inferring his or her interest profile. This mainly depends on the number of followees which could be linked to an entity and the quality of that entity linking. We evaluated both issues on a sample dataset.

5.3.1 *Method and Sample Description*

We conducted experimental research by crawling the profiles of 3000 Twitter accounts (with over 350 000 followees in total) chosen randomly from an updated dataset based on [5, 105]. Afterwards we analyzed the number of followees that could be linked to an entity and assessed the quality of that entity linking by applying the disambiguation heuristics mentioned in the second step of section 5.2 on page 69.

3 http://xmlns.com/foaf/spec/ – last accessed March 2020

4 http://smiy.sourceforge.net/wi/versions/20100812/spec/weightedinterests.html – last accessed March 2020

A first analysis of the sample showed that over 72 % of the users in the sample are friends with more than 50 other accounts. More than half of the Twitter accounts examined had between 50 and 200 followees. The overwhelming majority (91 %) used the English language version of Twitter.

5.3.2 *Quantitative Results*

For analyzing the number of followees that could be linked to a corresponding Wikipedia page entity we used the MediaWiki Web API[2]. As this API allows search on the English Wikipedia with an auto suggest feature enabled or disabled, we did the calculation for both. Table 5.1 and table 5.2 on the facing page show the results for different selections on the sample. The numbers include the shares of followees which could be linked to an entity unambiguously, the followees that could be linked to more than one page (ambiguity) and the followees that could not be linked to any entity at all.

Table 5.1: Quantitative results (auto suggest enabled).

	followees in % linked		
Selection	unambig-uously	ambig-uously	not at all
None	69.89	7.14	22.72
Number of followees > 50	71,08	7.11	21.65
Number of followees < 50	66.77	7.23	25.51
English language version	71.24	7.24	21.27
Other language version	54.84	6.05	38.87
English language version, number of followees > 50	72.44	7.20	20.22

On average about 70 % of the total number of followees could be linked unambiguously to an entity by the MediaWiki API with the auto suggest feature enabled. In less than every tenth case (7.14 %) more than one disambiguation (articles of the same name) was possible. About a fifth of the followees could not be linked to any entity even with the auto suggest feature enabled. Considering only accounts using the English language version the share of followees linked unambiguously is significantly higher (71.24 %) than with other language versions (54.84 %). The same effect, even though to a lesser extent, can be seen when comparing accounts that have more and less than 50 followees. The best success rate (72.44 %) is achieved by a combined selection

of accounts using the English language version of Twitter with more than 50 followees.

Table 5.2: Quantitative results (auto suggest disabled).

	followees in % linked		
Selection	unambig-uously	ambig-uously	not at all
None	41.23	5.73	52.93
Number of followees > 50	42.74	5.81	51.35
Number of followees < 50	37.24	5.54	57.08
English language version	42.61	5.88	51.39
Other language version	25.84	4.06	69.99
English language version, number of followees > 50	44.17	5.95	49.79

With the auto suggest feature disabled, the share of followees that could be linked to an entity is, as one could expect, lower (41.23 % compared to 69.89 %). However the trends for the different sections are very similar. For accounts with more than 50 followees that use the English language version barely half could be linked to an entity (6 % of these ambiguously).

5.3.3 *Qualitative Results*

The quantitative results may not necessarily imply that the quality of the entity linking is sufficient. This depends on whether the followee was linked with the semantically correct entity. For instance "common" people that share the name with a celebrity coincidentally might be linked to a Wikipedia page. To assess the quality of the entity linking we applied some of the disambiguation heuristics mentioned in section 5.2 on page 69:

5.3.3.1 *Overlap Coefficient*

Even though the applicability for tweets might be limited due to their short length and informal character we first calculated the overlap coefficient as a simple syntactic measure for assessing the link quality. This was done by collecting the last 20 tweets for 7500 randomly chosen Twitter users that could be linked to a Wikipedia page by the MediaWiki Web API[2] (auto suggest enabled) and the summary of the linked page (usually the very first section). Afterwards we tokenized the crawled input and converted it into a set of words, which also

removed duplicates. On that basis we calculated the overlap coefficient as shown in equation (5.2) (where X and Y are the two word token sets compared).

$$overlap(X, Y) = \frac{|X \cap Y|}{min(|X|, |Y|)} \tag{5.2}$$

The overlap coefficient was calculated for both, a random mapping of tweets and page summaries (the baseline) and for the linking suggested by the MediaWiki Web API. This was done before a text normalization (stop word removal and stemming) was applied as wells as afterwards. With text normalization the results show (see table 5.3) that the mean overlap coefficient for the entity linking is twice as high as for the random baseline mapping (Cohen's $d = 0.47$). Without text

Table 5.3: Qualitative results (overlap coefficient).

	Entity Linking $n = 7500$		Baseline $n = 7500$	
	M	SD	M	SD
no normalization	0.2325	0.0910	0.2168	0.0871
normalization	0.0609	0.0643	0.0369	0.0365

M: mean, SD: standard deviation

normalization the effect (Cohen's $d = 0.18$) is clearly smaller. All value differences were highly statistically significant ($p < 0.001$).

5.3.3.2 *Reverse Linking*

A very easy way to get a quick estimation of the entity linking quality is to search on Twitter for accounts with the name of the Wikipedia page title (entity). By doing this reverse linking we ended up in about 80 % of the cases with the account we started the entity linking from. Both Twitter and Wikipedia have optimized search indices and the article title sometimes contains additional disambiguation information (e.g. "footballer" for "Thomas Müller"). A high success rate could be seen as indicator of a good entity linking, but as the search algorithms and indices of Twitter and Wikipedia are black boxes for us this could only be a first clue.

5.3.3.3 *Sense Prior*

Sense Prior is a probabilistic approach which assumes that the most frequent word meaning dominates the others [155]. For that purpose the relative frequencies of so-called surface forms linking to an entity

are calculated and the most frequent one is assumed to be the correct disambiguation. In this evaluation we used a dataset based on the internal link structure of the English Wikipedia to calculate the frequencies. Let $l = (s, a)$ be an internal link which points to article a with the link text (surface form) s and let $n(s)$ describe the number of occurrences of that surface form in all articles then

$$P(a|s) = \frac{l(s,a)}{n(s)} \tag{5.3}$$

is the probability that entity a is the correct disambiguation for surface form s.

We calculated that probability for 10 000 randomly chosen followee names (the surface form in this case) of our sample (no auto suggest, English language version and more than 50 followees) and compared the entity with the highest probability to the linked entity. If the Sense Prior dataset could provide a disambiguation (the case in 78 %) it corresponded with a probability of over 90 % with the linked entity.

5.3.4 *Analysis and Discussion*

The results of our empirical research show that without auto suggest almost half of the followees and with auto suggest over two thirds of the accounts a user is following could be linked to Wikipedia page entities successfully. This implies that the Twitter followees of a user actually could be a sufficient and broad basis for inferring interest profiles. As ambiguity does occur only in about one out of ten cases it should have little effect. The qualitative evaluation points towards the same direction: With an overlap coefficient twice as high as for a random baseline mapping and success rates of about 80% for the reverse linking and around 80 % and 90 % for the probabilistic disambiguation heuristics the entity linking quality could be seen as sufficient as well.

We conclude, that the Twitter followees of a user provide already a sufficient input, both quantitatively and qualitatively, for inferring meaningful interest profiles.

5.4 USER STUDY

Even though the groundwork in the last section showed that the Twitter followees are a sufficient base for inferring interest profiles, it does not guarantee high quality of the resulting profiles. The evaluation of personalization and/or recommendation systems typically involves a user study [91] in order to assess the profile quality. For that purpose we implemented the approach presented in section 5.2 on page 69 in Python and evaluated it with real users.

5.4.1 *Experimental Setup*

For our evaluation we generated four different profile types defined by the number of spreading activation iterations, the decay factor and the application of disambiguation heuristics (see table 5.4). After the

Table 5.4: Evaluated profile types.

	Iterations	**Decay**	**Disambiguation**
Profile type 1	5	0.2	No
Profile type 2	5	0.2	Yes
Recommendations	3	0.2	Yes
Comparative Evaluation	5	0.2	Yes

users had registered by providing their Twitter screenname and e-mail, they were notified by a mail providing a link to their personalized questionnaire. This questionnaire had four pages that corresponded with the four different interest profile types shown in table 5.4.

On the first page the user was presented the top 20 interest categories (most weighted nodes) of the first profile type. The participants were asked to indicate their strength of interest for each category on a four-point Likert scale ranging from "very interesting" to "not interesting at all". The second page was pretty much the same presenting the top 20 interests of profile type 2 that mainly differed in whether disambiguation heuristics were applied or not. On the third page five Wikipedia articles that were assigned to the interest categories of the third interest profile (a smaller number of iterations was used to get more specific results) were shown. Again the participants were asked to indicate their strength of interest in the topics covered by these articles. The last page showed the users ten interest categories randomly picked from other users' profiles. As the categories did not appear in their interest profiles, no interest of the users in these categories was assumed and they were asked to evaluate whether this was correct or not.

Afterwards the participants were provided with a link they were asked to send a friend of theirs. This link lead to a one-paged survey that presented the user's friend with 20 interest categories. One half consisted of the top 10 interests of profile type 2 and the other half were the randomly picked interest categories that our approach assumed to be not interesting. Now the user's friend was asked to evaluate the interest of his or her friend in these interest categories. Following [223] these answers were used to compare the performance of the friend and our algorithm in predicting the user's interests. Whereas the pages one and two were obligatory the last two steps could be skipped by the participants.

5.4.2 *Sample Description*

During the evaluation period from 30 June to 10 July 2015 64 Twitter users registered for the user study and 52 of them completed the survey (response rate of 81.25 %). A participant had 205 followees on average, while the median (114) was considerably lower. The used Twitter language versions were half German and half English. Barely half of the users posted fewer than 100 tweets (over 15 % nothing), which means that approaches based on the tweets would fail to generate interest profiles for that users. 46 participants submitted the optional third page and 17 people took part in the fourth step (comparative evaluation).

5.4.3 *Results*

5.4.3.1 *Evaluation of Likert Scale Items*

The possible answers of the Likert scale were encoded with values ranging from 1 for "not interesting at all" to 4 for "very interesting". Whereas the top 20 interests of profile type 1 scored 2.38 ± 0.33, the same selection of interest categories for profile type 2 scored higher with 2.80 ± 0.39. This trend could be found for all n-best selections (see table 5.5) reaching a maximum difference of 0.7 for the top 5 interests. The recommended Wikipedia articles (profile type *recommendations*) have been evaluated with an average score of 2.46 ± 0.33 by the participants.

Table 5.5: Mean scores of Likert scale items.

	Type 1 $n = 52$		Type 2 $n = 52$	
	M	SD	M	SD
Top 5 interests	2.38	0.33	3.08	0.44
Top 10 interests	2.39	0.33	2.95	0.42
Top 15 interests	2.39	0.33	2.86	0.41
Top 20 interests	2.38	0.33	2.80	0.40

5.4.3.2 *Precision*

The precision measures the ratio of relevant recommended items to all recommended items. For calculating the precision we considered

items rated as "very interesting" and "interesting" as relevant to the user (true positive) and items rated as "hardly interesting" and "not interesting at all" were considered irrelevant (false positive). Figure 5.2 depicts the precision curves for different n-best selections of profile type 1 (red curve, dashed) and profile type 2 (blue curve, solid). Again

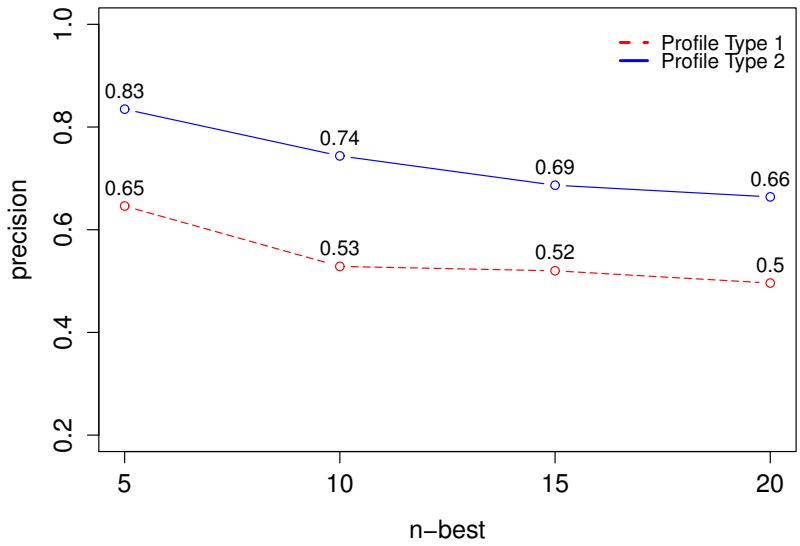


Figure 5.2: Precision curves profile type 1 and 2.

profile type 2 (disambiguation heuristics applied) dominates profile type 1 (no disambiguation) in each n-best selection: Regarding the top 5 interests, users indicated a correct assignment for over 80%. For all inferred topics (top 20), at least two thirds are considered relevant by the users. Similarly, two of the top 3 Wikipedia articles recommended in the third step (profile type *recommendations*) are considered relevant.

5.4.3.3 *MAP and MRR*

Mean Average Precision and Mean Reciprocal Rank answer the question of how well the interests are ranked at top-k and how early relevant results appear [125]. Again both MAP and MRR scored higher for profile type 2 (0.72 and 0.85) than for profile type 1 (0.50 and 0.68).

5.4.3.4 *Comparative Evaluation with a User's Friend*

Profile type 4 was built up with top-10 interest categories of profile type 2 and 10 interest categories where no interest was assumed. The users were asked to evaluate this profile and send a link to a friend of theirs to do the same. The performance of our algorithm and the friend's assessment was compared by the user's evaluation (benchmark). Table 5.6 on the facing page shows the confusion matrix comparing the performance of the algorithm introduced in this paper and the user's friends (in brackets). With a combined success rate of 74 % versus 60 % our approach clearly outperforms the friend in

Table 5.6: Performance algorithm and friend (in brackets).

	Recommended	**Not recommended**
Relevant	73 % (55 %)	27 % (55 %)
Not Relevant	24 % (35 %)	76 % (65 %)

predicting the user's interests. Differences in the above mean values are statistically significant ($p < 0.01$).

5.4.4 *Analysis and Discussion*

The results of the user study show that a user's followees are not only a sufficiently broad basis for inferring interest profiles but these interest profiles are also a valid representation of the user's interests. Profile type 2 scored better than profile type 1 in all quality measures calculated. This implies that the disambiguation heuristics have a significant impact on the quality of the generated interest profiles. With over 7 out of 10 items being relevant to the users our approach could achieve state of the art results and performed even better in predicting the users' interests than their friends (thus humans) did.

5.5 PRODUCTION VERSUS CONSUMPTION BASED PROFILES

We did additional research to answer the introductory question, which also was raised by [185], of how interest profiles based on the user's tweets (production) and followees (consumption) differ. Research on this questions also taps into the long-running debate on consumption vs. production online, where it is often argued that these two actions, particularly with regard to social media and digital content, are inseparable.

5.5.1 *Approach*

To provide the basis for a valid comparison of the two different profiles we generated them using the same approach and knowledge base, but extracted the entities from the user's tweets in one case and from her followees in the other. However, we did not conduct a second user study to assess and compare the quality of the two different profile types, but only compare the created profiles.

In the first stage we extracted the entities for the consumption based profile of the user's friend list as described in section 5.2, whereas for the production based profile the Illinois Wikifier [151], an external tool, was used to extract entities from the user's tweets.

In the subsequent stage, which was the same for both profile types regardless of their input, a spreading activation algorithm was performed on the Wikipedia Bitaxonomy [54] to aggregate the single interest entities to a more abstract and broader interest profile.

Finally, the interest profiles, which represent the user's interests as a list of weighted Wikipedia categories are used to calculate the cosine similarity (as shown in equation (5.4)) of the two different interest profiles.

$$sim(X, Y) = cos(\theta) = \frac{X \cdot Y}{||X||\ ||Y||} \tag{5.4}$$

where X is a vector representing the interest items' weights of the production based profile and Y is the corresponding vector for the consumption based profile. As we use a taxonomy as knowledge base this allows us the comparison of the profiles on different levels of abstractness represented by varying parameters of the spreading activation algorithm (e.g. number of iterations or the decay factor).

5.5.2 *Sample*

To calculate the similarity of the two different profile types we selected a random sample of 50 Twitter users from the Twitter sample endpoint, which allows to access a small sample of all public statuses and applied a set of selection criteria (e.g. only public accounts, a sufficient number of tweets and friends) on it to get suitable accounts for our experiment only. Apart from that, we took the ten participants from the user study described in section 5.4 that rated the interest profile best and the ten that rated it worst. This leads to a total number of 70 Twitter users for which both profile types and their cosine similarity were calculated.

5.5.3 *Results*

Compared to the followees of a user it appears to be easier to extract entities from the tweets as results showed that about 2.5 times more entities could be extracted using tweets rather than friends as input. Only about 2 % of the extracted entities (meaning Wikipedia pages) were shared of both sets. Whereas an intersection of 9 % could be found for the first level categories (representing the initially activated nodes). At the first glance these results indicate that the generated profiles and inferred interests do not seem to be too similar.

To gain a deeper insight, the cosine similarity of the two interest profiles (list of interest categories and their corresponding weight) was calculated with a fixed decay factor (0.1, 0.2 and 0.3) and iterations ranging from 1 to 20. The number of iterations represents the different levels of abstraction, ranging from very concrete (one iteration meaning the initially activated nodes and their corresponding weight)

to very abstract (20 iterations upwards in the Wikipedia category taxonomy).

As shown in figure 5.3 the cosine similarity of the concrete profiles (one iteration) comprising the weighted initially activated nodes is with about 0.66 higher as it would be expected given the small intersection of entities and categories. With an increasing number of

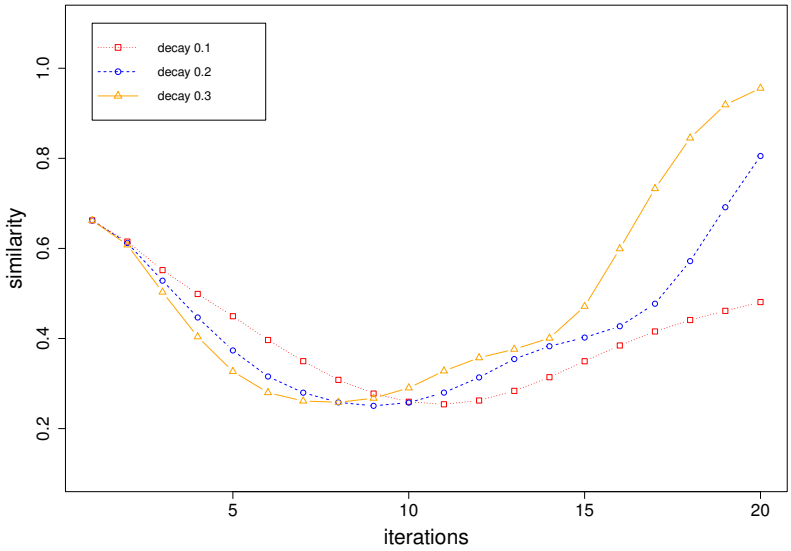


Figure 5.3: Cosine similarity of production and consumption based profiles for different decay factors.

iterations the similarity of both profiles is decreasing, whereby the minimum is reached later if the decay factor is smaller. As expected the similarity is rising for a high number of iterations again, as the categories reached by the spreading activation algorithm now are of a very abstract nature, i.e., the spreading activation accumulates in the top level categories. In general, profiles appear to be more similar on very concrete and abstract levels of the taxonomy.

Figure 5.4 on the next page compares the similarity of the 10 profiles that have been evaluated worst and the 10 that have been evaluated best in the user study that is described in section 5.4 on page 75. The overall trends and the "u"-shape that can be seen in similarity with an increasing number of iterations are the same for both selections. However, the worst evaluated profiles have a higher similarity on a more concrete level (smaller number of iterations) whereas the best evaluated profiles seem to be more similar on an abstract level (higher number of iterations) approaching complete similarity at 20 iterations.

5.5.4 *Analysis and Discussion*

In general the cosine similarity of both profiles turns out to follow a "u-shape" along an increasing number of iterations. This means that

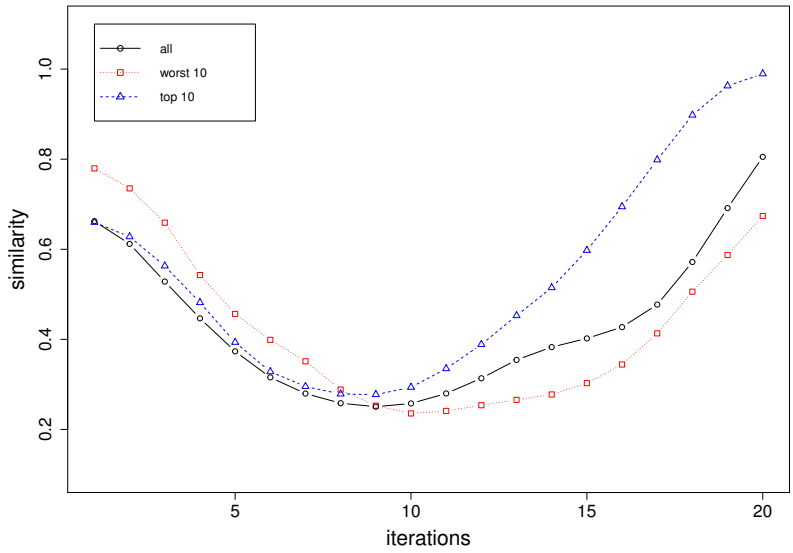


Figure 5.4: Cosine similarity of production and consumption for 10 best and worst evaluated profiles.

profiles are more similar on very concrete and abstract levels of the Wikipedia Bitaxonomy (the used knowledge base). While the latter is expectable, since the spreading activation accumulates on the top level categories, the former is an interesting finding. In particular, since the profiles only share 2% of extracted entities. The larger intersection on the first level of categories (9%) is also not surprising, as the categories provide an abstraction of the very specific entities. That is, the number of (available) categories is smaller than the number of (available) entities. However, the high cosine similarity is an interesting finding, which suggests, that even though the intersection of entities and first level categories is low, the weights accumulate in the same categories. Still, the huge amount of categories in which the approaches differ seem to trigger different routes through the category graph to the top levels.

Even though there are differences in the concrete entities that could be extracted from the different sources of input, the information we get based on the consumption and production of the users' Twitter accounts are quite similar for certain levels of abstraction. The results therefore support the hypothesis that consumption and production online, particularly with regard to social media and digital content, are inseparable.

5.6 SUMMARY & CONCLUSION

In this chapter we introduced an approach for inferring semantically meaningful interest profiles from the accounts a user follows on Twitter. Because the followees are the only input used, it is possible to

generate interest profiles even for users that post no tweets. Also, investigating the coverage of followee lists in terms of named entities in Wikipedia revealed that followees indeed provide sufficient input for creating comprehensive semantic interest profiles. As passive social media use is on the rise, the approach is an important contribution to the development of future social-media-based recommender systems that try to address the cold start problem. By conducting an extensive user study we could show that our approach achieved state of the art (and superhuman) results in predicting a user's interests. A comparison of followee- and tweet-based profiles revealed high similarity on very concrete and abstract levels, suggesting that passive and active use are tightly coupled.

Given the high quality of interest profiles extracted based on a user's followees, we conclude that the presented approach can complement the profiles defined in section 4.6 on page 62. While those profiles are generated from the query history, the followee based profiles add an additional dimension by using social media as information source. This "external" information source also alleviates the cold start problem for our personalization strategy. Spreading activation is straightforwardly applied to named entities that form the query, which allows a seamless integration.

Part III

PRESENTATION OF RESULTS

RQ 2

How to adapt result presentation to zero-effort queries?

6 INTRODUCTION

When a user actively searches for information, she expects results to be presented after issuing a query. However the story is different in our zero-effort query scenario: Since the query process is fully automatic, we present results proactively. Therefore, the user may be presented with results, even though she is not interested in them and does not want to be interrupted in her current task.

Also, our scenario is different from many other zero-effort query scenarios. In most of them, the user actively indicates that she *has* an information need, without *expressing* this information need. This active indication comprises for example opening a search engine or a restaurant suggestion app. In both examples it is obvious that the user *has* an information need, i.e., she plans to search for *something* or is looking for a restaurant. In the latter example, the information need is even partially obvious (looking for a restaurant). In this case, a meaningful choice is to present nearby restaurants to the user. In the former example, the search engine could try to anticipate the user's information need and present results accordingly. In both cases, the user would not be bothered by the presentation of nearby restaurants or search results as she could easily ignore them and still express her actual information need.

As we present results proactively, without knowing whether the user *has* an information need, we need to carefully consider this circumstance. Results should be presented in an unobtrusive manner, allowing the user to easily ignore or not even recognize them at all. We investigate user interface issues related to this fact and evaluate different interface options to address it in the following chapter 7 on page 89.

Another issue that concerns result presentation and arises from the automatism of zero-effort queries is system transparency. If only the results are displayed without further explanation, the user has hardly to no chance to determine what information need the system assumes and how it came to that conclusion. In order to increase transparency, we propose to display the actual query along with the results, revealing (at least partially) the information need the system assumes. Further, we visualize the triadic relationship between context, query and results, providing clues to the user how the system arrived at this information need assumption and how the results are related to it. The details of our approach are described in chapter 8 on page 103.

A further issue that can be addressed in the presentation interface is the difficulty of recalling where and how information was found,

known as the "Lost in Hyperspace syndrome" [39]. While strictly speaking, the "Lost in Hyperspace syndrome" refers to the navigation of hypermedia only, the difficulty of recalling is also present in search. If the user does not actively store retrieved results or queries, recalling where or how information was found requires high recognition activities or may not even be possible anymore. This issue is even more prevalent when information is not retrieved actively by the user herself, but proactively by zero-effort queries (without involving manual user effort). To counter this fact, we present a search query history visualization that automatically stores the queries used to retrieve results and allows the user to navigate back in the query history. Details about this visualization called "QueryCrumbs" are provided in chapter 9 on page 109 which concludes this part.

7

PRESENTATION INTERFACE

In this chapter we identify and investigate user interface issues related to zero-effort queries of digital resources. Search is an integral part of users' activities on the Web and plenty of design guidelines exist for search user interfaces for when the user's primary goal is information retrieval. Typically, those interfaces consist of a part for query specification, where the searcher expresses an information need in the query format of the search system and the corresponding search results, most commonly displayed as a vertical list of result summaries [71].

Zero-effort queries eliminate the input interface for information need expression by retrieving potentially relevant resources automatically without explicit user interaction. Depending on the user's task and the quality of the context detection and retrieval mechanism, the presentation of the retrieved resources may or may not be desired by the user. Thus, we propose a two-stages approach: first, the user is notified about the retrieval of new resources, and second, the resources are presented. More specifically, we investigate the following core questions:

1. How can a user be visually notified that additional results are available?

2. With which user interface elements should the results be presented?

The first question relates to notification styles, and the second question relates to representation styles. Additionally, a transition from the notification to the result representation is required. This transition may be fully automatic and with no visible delay in the user interface, e.g., when the user is notified about the existence of new resources, the resources are automatically displayed alongside. Alternatively, the transition can be explicit or manual, e.g., the user needs to perform an action in the user interface to see the retrieved results.

In this chapter, we categorize interface issues for Web-based zero-effort queries, and identify different interface types. In order to define notification and interface styles we make the following assumptions:

- Focus on Web-based scenarios, i.e., results should be integrated in Web browser.

- Focus on visual interfaces, ignoring for instance, audio or haptic interfaces.

- Retrieval service is external to Web-application, i.e., there is no internal access to the current website (apart from JavaScript injection inside the browser).

Further, we present results of a survey-based user study to elicit user preferences for the derived interface styles.

The rest of the chapter is structured as follows: The next section presents related work on issues of search and awareness interface design relevant in a zero-effort query setting. The user interface considerations are described on a conceptual level in section 7.2 on page 92, deriving four notification styles and six presentation styles. Section 7.3 on page 94 then presents the survey design and results, followed by a discussion and implementation recommendations in section 7.4 on page 100 and the conclusion in section 7.5 on page 102.

7.1 RELATED WORK

In this section we review related work w.r.t to human information seeking models, design of search user interfaces and awareness interfaces.

7.1.1 *Models of Human Information Seeking:*

Andrei Broder proposed a taxonomy of Web search, classifying searches either as navigational, transactional, or informational [26]. Among these, informational Web search is the most common and assumes an initial information need of the searcher. Shneiderman's model of human information seeking encompasses four basic steps [182, 183] and has been extended to seven steps by Marchionini and White [126]: (i) recognition of information need, (ii) accepting challenge to take action, (iii) formulating the problem, (iv) expressing the information need in search system, (v) examination of results, (vi) reformulation, and (vii) usage of results. Mulhem and Nigay applied Norman's stages-of-action model [139] to information retrieval [136], identifying the stages: (i) goal or problem setting, (ii) intention (information need), (iii) action specification, (iv) execution of actions, (v) perception of system state, (vi) interpretation of system state, (vii) evaluation of the state with respect to initial goal. Common to these models is a stage or a set of stages that correspond to the formulation and *execution* of a search, and a stage or a set of stages that correspond to the *evaluation* of the received results. In zero-effort query settings, the execution step is automatically performed and replaced by a "notification of results" step for the user. Once the results are retrieved, the subsequent steps of evaluation are the same in traditional search and zero-effort queries. In terms of Norman's stages-of-action model [139] automatic retrieval is minimizing the gulf of execution, but might enlarge the gulf of evaluation if the automatically inferred query does not match the

initial user goal. In this chapter, we investigate interface guidelines for a browser-based user interface supporting human information seeking with a fully automatic query execution stage.

7.1.2 *Design of Search User Interfaces*

As a query in a zero-effort query setting is issued automatically, the primary focus of the search interface is on the presentation of search results. In general, nowadays search interfaces are (and should be) kept as simple as possible. Hearst identifies one of the main reasons therefor in search being "a means toward some other end, rather than a goal in itself. When a person is looking for information, they are usually engaged in some larger task, and do not want their flow of thought interrupted by an intrusive interface." [71]. This background nature of the search task becomes even stronger in a zero-effort query setting, when the search is performed automatically, emphasizing the simplicity claim. Hearst proposes the presentation of the search results as a vertical list, with so called document surrogates, containing summary information, such as document snippets, abstracts or metadata [126]. There is no distinct advice on the surrogate length, but Cutrell and Guan discovered, that more information in the surrogates improves informational tasks, while degrading performance for navigational tasks [41].

7.1.3 *Awareness Interfaces*

In order to keep distraction low, while making users aware of additional information, Rhodes introduced "ramping interfaces" [157], that convey information on different stages of granularity. The amount of information conveyed increases on higher levels, posing less cognitive load on the user in the early stages. The concept of awareness is also researched by Cadiz et al [29]. They present a system, that shows high-level information from various applications in a permanent sidebar and is capable of delivering notifications from specific applications. Yamada et al. [218] exploit human cognitive properties, such as visual field narrowing and in-attentional blindness. When a user concentrates on a task, the visual field narrows and changes outside are not recognized. Hence, positioning the awareness interface outside this area, allows for easily ignoring notifications when concentrating on a task, while recognizing them afterwards. According to McFarlane et al., “people are more interruptible for a brief signal that announces the existence of an interruption than they are for the full interruption itself” [129]. Therefore, we follow a two-staged approach, with notifications in the first and result presentation in the second stage.

7.2 USER INTERFACE STYLES

In zero-effort queries, the current user context is exploited to automatically retrieve relevant resources without explicit user interactions (cf. part ii on page 13). Figure 7.1 depicts the general process, ignoring

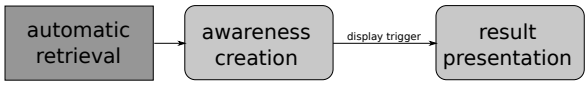


Figure 7.1: Overview of general procedure to present zero-effort query results to users.

the context detection and pre-processing steps necessary for automatic generation of a search query. Once resources are available, the user has to be notified of their availability. In this stage, the user does not need to get any details about the results, just the information, that there are results available. After the user has been notified, she might decide to view the result list, and perform some interaction in the user interface to indicate this decision (e.g., move the mouse over the notification item, or click on the item). The type of the transition from notification to presentation is denoted "display trigger" in figure 7.1. This means, we have to investigate the different possibilities for notifications, display trigger, and result presentation. The *Notification Interface Style* refers to the visual method presenting the availability of new resources to the users. The goal of the notification is to make users aware that new resources are available. The *Representation Interface Style* refers to the visual method presenting the actual resources. The building block for this is the document surrogate. The *Display Trigger Style* refers to the interaction possibilities for showing or updating the representation of results after a notification has been received. In the following these styles are discussed in more detail.

7.2.1 *Notification Interface Styles*

We restricted the notification styles to those which are implementable in current Web browsers and would work for arbitrary websites. Further, we only focused on visual notification styles, because audio for instance would require a not muted loudspeaker which can not be generally ensured. The notification should be unobtrusive, and require little space, since the encoded information is either binary (there are new results or not) or one number (the number of new results).

We identified four different notification interface styles, which are sketched in figure 7.2:

N-1. A notification icon appears in the browser task bar (see figure 7.2a).

N-2. The notification icon changes its appearance (see figure 7.2b).

N-3. A notification bubble appears in the browser task bar (see figure 7.2c).

N-4. A line appears within the current Web page (see figure 7.2d).

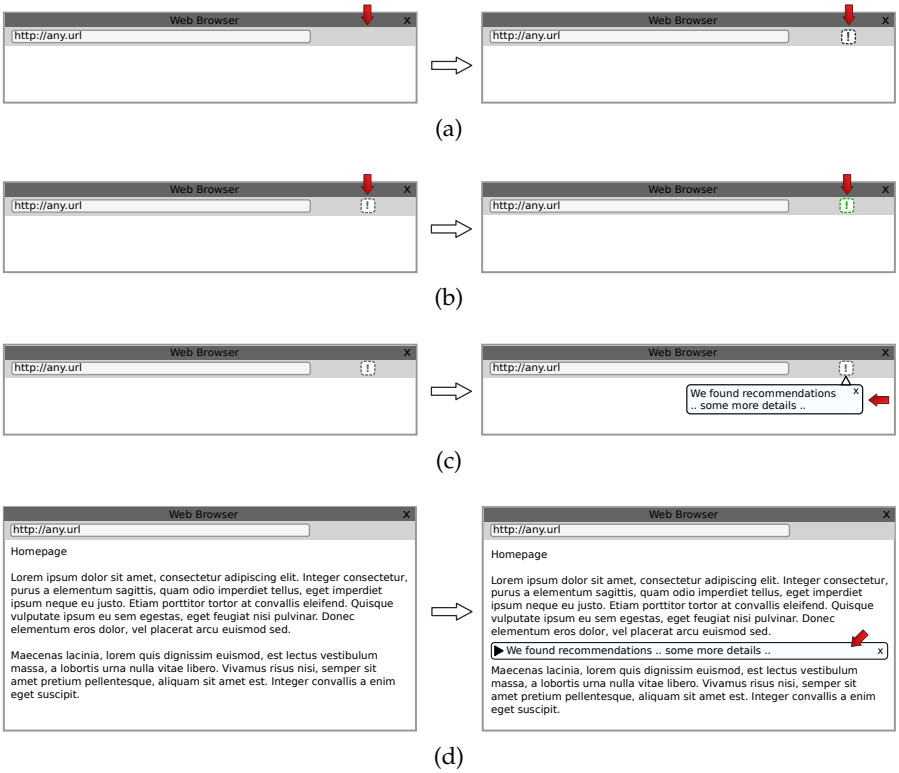


Figure 7.2: Notification interface styles. A notification (a) icon appears, (b) icon changes its appearance, (c) bubble appears, (d) line appears. The left column shows the interface before new results were discovered and the right column shows the interface after new results have been found.

7.2.2 *Representation Styles*

This style reflects the way in which the search results are represented. Because each result may be a larger document, not the actual content is displayed, but rather a summary, the so called document surrogate [126]. The minimal information contained in a document surrogate is the title of the document or label of the resource. Depending

on the application, additional information, so called faceted metadata, may be helpful, such as the author, origin or creation data. Further, a short description of the content may be available. But even with the shortest possible document surrogate (containing only the title), the representation of a result list requires more screen space than the notification. We identified six different representation interface styles, which are sketched in figure 7.3 on the facing page:

R-1. Results are displayed in a pop-up window (see figure 7.3a on the facing page).

R-2. Results are displayed using a split pane (see figure 7.3b on the facing page).

R-3. A new tab is opened for the results (see figure 7.3c on the facing page).

R-4. A box showing the results is inserted into the website (see figure 7.3d on the facing page).

R-5. A marginal note on the right-hand side of the page contains the results (see figure 7.3e on the facing page).

R-6. A marginal note on the bottom of the page contains the results (see figure 7.3f on the facing page).

7.2.3 *Display Trigger Styles*

Given a notification, the user may either decide to view the results, or the results may be automatically displayed, leading to two possible display trigger styles:

T-1. Explicit or manual, requiring a user interaction (click, mouse-over, key, key combination, menu selection),

T-2. Implicit or automatic, requiring no user interaction (pop-up with recommendation content, automatic update of recommendation display).

With an automatic transition, the notification and the presentation may be either indistinguishable for users (i.e., a notification icon appears and the results are displayed alongside) or the presentation of results is itself the notification (only showing the result list). Conceptually, however, notification and display trigger to presentation remain two different stages.

7.3 EVALUATION

With an online survey we wanted to evaluate whether users have a preferred notification and/or a preferred representation style for

Figure 7.3: Representation interface styles. Principle ways of injecting additional resources on a Web page. Results might appear (a) in a pop-up window, (b) a split pane, (c) a new browser tab, (d) inline the Web page, (e) in a margin note on the right, (f) in a margin note at the bottom.

recommendations. Knowing the preferences for notification and representation style, the design process for the application user interface can be guided already early in the development stage by only implementing the mostly preferred versions. If we additionally found a clear preference to one of the possibilities for the majority of users it would reduce the design and development effort for the interfaces even more. Concretely we wanted to test the following hypotheses:

H1 The majority of users has a clear preference towards one of the suggested *notification* interface styles.

H2 The majority of users has a clear preference towards one of the suggested *representation* interface styles.

Additionally, we wanted to assess whether users have additional suggestions for notification or presentation styles.

We further wanted to evaluate whether there is a difference between participants working with Web-based learning management systems (e.g. moodle[1]) and the general public. In learning management systems, relevant resources, which are retrieved automatically are of interest for users, who would like to investigate the learning topic more deeply and beyond the provided content. We assumed, that the requirements for displaying zero-effort query content in learning management systems differ from those in the general Web browser setting.

Therefore, we distributed two identical copies of the online survey, one to people working with learning management systems (developers, managers, project coordinators and teachers) and one for the general public. We did not evaluate preferences for display trigger styles in the survey, for two reasons. First, we wanted to keep the survey short (completion time below 5 minutes). Second, we found in a pre-test that intent behind the question was hard to convey to users and led to wrong answers. This led us to the conclusion that the survey form is not appropriate for eliciting preference on trigger styles.

7.3.1 *Test Material*

The survey was implemented using the open source survey application Limesurvey[2]. The questions were prepared in two languages, English and German. The survey's core questions relate to the above introduced hypotheses. The question related to hypothesis **H1** was:

> How would you like to be notified if relevant resources were available?

1 https://moodle.org/ – last accessed March 2020
2 https://www.limesurvey.org/ – last accessed March 2020

The answer possibilities are introduced in section 7.2.1 on page 92, multiple choices were possible. The question related to hypothesis **H2** was:

> In which principle way would you like the relevant resources to be presented? The following shows example sketches. Please note, that the number of recommendations is arbitrary as well as the information shown for each recommendation.

The answer possibilities are introduced in section 7.2.2 on page 93, multiple choices were possible. For both questions, interface mock-ups were presented to users for better understanding of the presented choices.

7.3.2 *Procedure*

The survey was distributed via mailing lists and at scientific conferences. Potential participants were given the survey URL and needed to register with their email address. They then received a token and could access the survey. By requiring an email address we tried to ensure that each participant could answer the survey only once. Although we could not guarantee that single users did not register with multiple email addresses this procedure was the best with respect to both, anonymity of users and uniqueness of responses.

The survey was structured as follows: First we presented a short introduction summarizing the goal of the survey and framing the context. Framing the context was implemented by setting up the following scenario (the mentioned screenshot is omitted):

> For answering the next two questions imagine the following scenario. You are browsing the Web with your favorite browser (Internet Explorer, Firefox, Chrome,..). For some of the visited websites interesting, additional information is found by the software. See the following screenshots for an example showing the Wikipedia page for "loom" and related cultural resources.

Then, the question about preferred notification style was asked, followed by the question about the preferred presentation style. Finally, the survey closed with questions about demographic data, such as age, gender, occupation, nationality and the preferred browser.

7.3.3 *Participants*

We received 75 completed responses, composed of 61 from the general public subgroup and 14 from the educational staff (see also table 7.1 on the following page). 34 (45%) categorized themselves as female

and 41 (55%) as male. 43 (58%) surveys were completed in German language and 32 (42%) in English. The age distribution is shown in figure 7.4, note that the age categories were predefined as shown in the figure. The majority of users (61%) were between 20 and 39 years old.

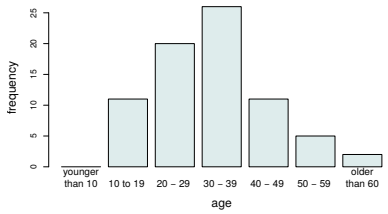


Figure 7.4: Age distribution of participants.

The responses to the multiple choice question about the preferred browser version are depicted in figure 7.5, revealing that Chrome and Firefox were the mostly used browser, while users rarely used Opera.

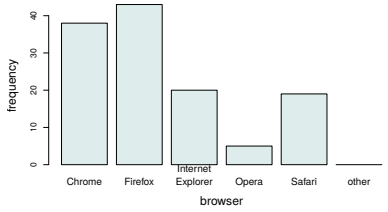


Figure 7.5: Browser distribution of participants. Multiple preferred browsers could be chosen.

7.3.4 *Results*

Table 7.1 summarizes the meta information of the survey responses. In total, 89 users requested a survey token, out of which 75 (84%)

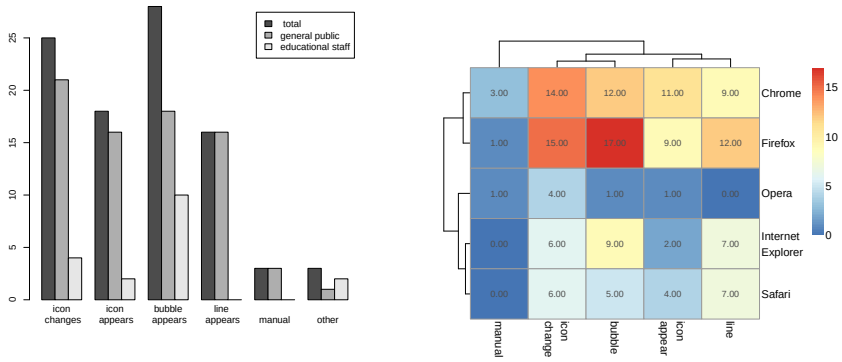
Table 7.1: Response overview for both user groups.

	Educational	**General**	**Total**
duration [months]	2.5	1	
avg. interview time	205 s	214 s	
total responses	19	70	89
full responses	14	61	75
completion rate	74%	87%	84%

completed the survey. The survey took approx. $3\frac{1}{2}$ minutes to complete on average. The survey was available over 4 months, the duration reported in the table refers to the time difference between the first and the last response.

7.3.4.1 *Notification Style*

Figure 7.6a summarizes the results for the notification styles. For the total population, the most preferable notification would be the appearance of a notification bubble (28 positive responses), followed by change of icon's appearance (25 positive responses) and the appearance of the icon (18 positive responses). Only 3 users would like to manually request resources. The results are slightly different for general public users, mostly they preferred the change of the icon's appearance (25 positive responses) followed by the appearance of a bubble (18), and icon or line appearance (16 positive responses each). On the contrary, the educational staff had a clear preference for the appearance of a bubble (10 positive responses).



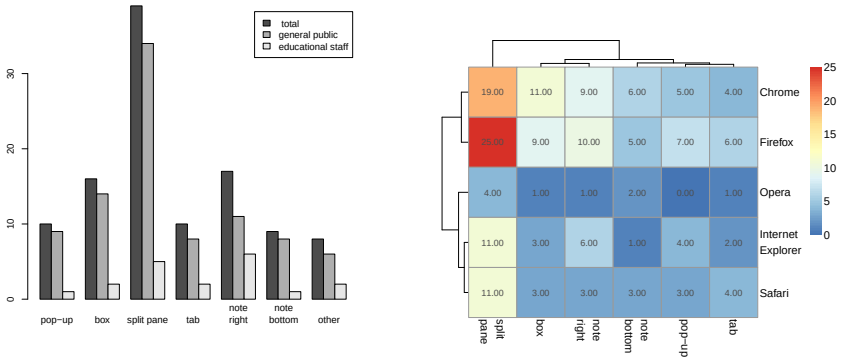
(a) (b)

Figure 7.6: Results for preferred notification styles (a), and relation to the preferred browser (b). Both questions allowed multiple answers.

7.3.4.2 *Representation Style*

Figure 7.7a on the next page summarizes the responses for the preferred representation styles. In total, the preferences are more distinct for the representation style than for notification style. The majority of users (39 positive responses), preferred a split pane like representation, while the other alternatives received 62 positive responses in total (excluding "other"). The tendency is the same for general public users with 34 positive responses for the split pane like representation. On the contrary, the educational staff preferred the style with a marginal note on the right (6 positive responses).

The free-text input for "other" suggestions contained answers like "drop down box" (once), "dialog box over the relevant text" (twice), and "separate split pane (if it is expandable and collapsible)" (three times). The latter is similar to the split pane type, but requests additional interaction possibilities, which were not indicated in the survey questions.



(a) (b)

Figure 7.7: Results for preferred representation styles (a), and relation to the preferred browser (b). Both questions allowed multiple answers.

7.3.4.3 *Relation of Styles and Browsers*

Figure 7.6b on the preceding page and figure 7.7b show an overview heatmap of the relation between preferred notification styles and preferred browsers. The numbers in the figure represent positive responses. Note that both questions in the survey were multiple choice questions, thus a survey of one user could potentially contribute to multiple squares in the heatmap. In the margin of both figures the cluster tree is shown, calculated using hierarchical agglomerative clustering on the similarity of row and column vectors, respectively.

For the relation of browser and notification style, it can be seen in figure 7.6b on the preceding page, that users who use Firefox and Chrome also have similar preferences in notification style, bubble appearance and notification icon change being the most preferred ones. Similarly users of Internet Explorer and Safari tend to prefer an appearing bubble or line, or a change of the icon's appearance. The most prominent combination across all browsers and notification styles is the Firefox/Chrome and bubble appearance/icon change (corresponding to the orange and red area in the figure).

There is less variance for the relation of browser and representation style (cf. figure 7.7b). Across all browsers users prefer the split pane style (corresponding to the left column), with the other styles being much less preferred across all browsers.

7.4 DISCUSSION

The survey revealed a strong user preference for the representation style (supporting hypothesis **H2**) and multiple equally important preferences for notification styles (not supporting hypothesis **H1**). The preferred representation style is the split pane, which has the advantage that it does not overlay current Web page content, and

no scrolling is required on the page to view its content. The three preferred notification styles are the appearance of a bubble, and icon appearance or change. The latter two can be implemented rather easily in a similar fashion, since an appearing icon can be simulated with an icon changing its appearance from invisible (or 100% transparent) to visible.

From the survey, we can derive two suggestions for designing user interfaces for Web-based zero-effort queries. First, we suggest to implement three different notification styles, and let users select their own preferred style. The three styles are a bubble-like notification, an icon which appears and an icon which changes its appearance on retrieval of new results. If the implementation resources are limited only implement the icon change/appear styles, which are technically very similar. The second suggestion is to implement a collapsible and expandable split pane for presenting results.

However, for the implementation of a split pane, the original page layout (in terms of CSS definitions) has to be taken into account. In the beginning of this chapter, we made the assumption that the "retrieval service is external to web-application, i.e., there is no internal access to the current website (apart from JavaScript injection inside the browser)". During implementing the split pane, we discovered that this requirement is not sufficient when working with *arbitrary* pages. The split pane alters the layout of the original page when injected via JavaScript which can cause issues up to rendering the original page unusable. During the time of the study, the extension mechanism of Firefox also allowed to alter the browser interface itself, circumventing this problem. That possibility has since then been removed in version 57[3]. Most of the major current browsers (according to market share[4]) share similar extension capabilities. These build on common Web technologies (HTML, CSS and JavaScript) and allow to inject code to the current Web page and predefined UI interactions/display options. Safari is the only exception which abandoned this approach in version 13[5] and uses dedicated apps[6] instead.

Hence, due to technical limitations in the majority of browser, we suggest to implement the split pane interface only when the developer is in control of the original page layout (i.e., the page for which related resources are presented). On arbitrary pages, there is no guarantee to keep the original layout intact. However, the options "pop up" and "margin note" (right or bottom of the page) are technically similar.

3 https://www.mozilla.org/en-US/firefox/57.0/releasenotes/ – last accessed November 2019

4 https://www.w3counter.com/globalstats.php?year=2019&month=10 – last accessed November 2019

5 https://developer.apple.com/documentation/safari_release_notes/safari_13_release_notes – last accessed November 2019

6 https://developer.apple.com/documentation/safariservices/safari_app_extensions – last accessed November 2019

Therefore we suggest to implement these and providing the user the option to switch between those three options. The accumulated positive responses for these three options (36) almost match the positive responses for the split pane (39). Further, the pop up may be implemented customizable by the user in terms of position and size.

7.5 CONCLUSION

In this chapter we investigated user interface considerations for web-based zero-effort queries. We proposed an interface approach with two stages, notification about new results and representation of the results. Further, we identified four different notification styles, six different representation interface styles, and two different possible result display trigger styles for a Web-based setting. In an online survey with 75 participants we elicited user preferences for the notification and representation styles. From the results and taking technical limitations into account, we derived the following interface guidelines for Web-based zero-effort queries on *arbitrary* Web pages:

- Implement three different notification styles and let users select their own preferred style, a bubble-like notification, an icon which appears and an icon which changes its appearance on retrieval of new results.
- Implement a user-customizable pop up (in terms of position/-size) with the option to transform it into a margin note at the bottom or right-hand of the page. Again, let users select their own preferred style.

8

(VISUAL) EXPLANATION

In regular search interfaces, the user explicitly expresses her information by formulating a query and is presented with corresponding results, retrieved by this particular query. A common feature in document retrieval is to highlight the occurrence of query terms in the document surrogate that has been retrieved by the query [71]. According to Hearst [71], this feature helps to draw the user's attention to parts of the document relevant to the query and shows how closely the terms appear to each other. The same author emphasizes on query term proximity as a strong indicator of relevance. Hence, the relatedness of a document to the query terms and the relevance to her information need can be easily assessed by the user.

In zero-effort queries, this relatedness and relevance is less obvious to the user. The (assumed) information need and corresponding query formulation is an automatic process. If only the accordingly retrieved results are presented, the user has no information about the information need assumed by the system, nor about the formulated query. In order to make the zero-effort query process transparent to the user we suggest to display the generated query alongside with a visual explanation of the triadic relationship between context, query and results.

In this chapter, we investigate whether users can benefit from making this relationship explicit and if so, in which way. We identify effectiveness and efficiency as aims of our visual explanation by a discussion of related work in the following section. We then discuss our approach to increase transparency, followed by the evaluation and discussion of the aforementioned aims.

8.1 RELATED WORK

In terms of explanations, zero-effort queries are similar to recommender systems: While zero-effort queries retrieve related results automatically, recommender systems seek items, that are not necessarily related, but of high interest to the user. That is, recommender systems aim to predict the rating or preference for items and present the highest scoring to the user. Explanations are part of a (demanded) shift from algorithmic evaluation (in terms of accuracy or precision) towards user-centric evaluation [102, 130]. Still, the majority of current research is focused on algorithms and offline evaluations, i.e., evaluations that do not include a human user [98]. Tintarev and Masthoff define seven aims of explanation in recommender systems [203]:

TRANSPARENCY explaining how the system works

EFFICIENCY helping users make decisions faster

SATISFACTION increasing the ease of usability or enjoyment

EFFECTIVENESS helping users make good decisions

PERSUASIVENESS convincing users to try or buy

SCRUTABILITY allowing users to tell the system it is wrong

TRUST increasing users' confidence in the system

The same authors later argue that "these aims can be incompatible, so any evaluation needs to state which aim is being investigated" [204]. They mention that "effectiveness may increase trust" as an example for complementary aims and "persuasiveness may decrease effectiveness" for contradictory aims. Gedikli et al. [57] investigated the impact of different explanation types on the first five aims in the above list. Their analysis revealed a strong relationship between transparency and satisfaction. Swearingen and Sinha identify transparency as a crucial factor for trust [186, 194]. The above-mentioned aims are also relevant to zero-effort queries, except for persuasiveness (at least in our scenario): As we do not aim to sell items, there is no need to persuade the user of a particular item.

Explanations have recently also gained attraction in the area of Deep Learning (see [150] for a survey). Deep learning models showed successful in a variety of areas, but it often remains unclear, how these models arrive at their decisions. Explanations are means for a deep learning model to justify and explain the rationale of its decision [73]. To gain insights into a model's inner workings, primarily *visual* explanations are used, in order to unfold the model's "black-box" character [36]. The goal of unfolding the black-box character is shared with zero-effort query systems.

8.2 TRANSPARENCY AND SCRUTABILITY

In terms of transparency and scrutability, explanations in our system are limited to the process until a query is generated and sent to a search engine. The search engine is a black box to us, hence, as we do not have information about its internals, we also have no means to provide explanations on how the search engine ranked the results. Therefore, the parts for which we can provide explanations are context detection and query generation.

Regarding context detection (cf. section 4.3 on page 39 and section 4.4 on page 46), we visually outline extracted paragraphs with a dotted gray line, telling the user which paragraphs have been extracted by our approach. Further, we frame the paragraph deemed in

focus by our approach with a green line. Both of these visual clues aim at transparency, i.e., showing the user which paragraphs have been extracted and which is deemed in focus. We offer the possibility to adapt the focus paragraph, therefore providing the user the option to correct a paragraph wrongly deemed in focus (scrutability).

In order to increase transparency, alongside with the retrieved results, we also display the query that was used to retrieve those results. As general users may not be familiar with formal query syntax, we display the query as main topic and keywords, instead of showing the actual boolean query. This query can be modified by the user, e.g. by removing or adding keywords or changing the main topic. Therefore, the user has the option to correct the query in terms of missing or irrelevant keywords, addressing scrutability. While displaying the query shows the user *which* query was used to retrieve results, it does not tell the user *how* the system arrived at this query. We address this issue by visual explanations described in the upcoming section.

8.3 VISUALIZING THE TRIADIC RELATIONSHIP BETWEEN CONTEXT, QUERY AND RESULTS

Zero-effort queries retrieve results for the current user context automatically without explicit user action. If those results are displayed without further explanation, the user does not gain any insights on how the results are related to the context or even what the system deems as relevant context. We partially address this lack of transparency by providing visual feedback for the current context (focus paragraph) and displaying the query as mentioned in the previous section.

Still, it may not be obvious, how the query relates to the context, i.e., *how* the query was constructed, nor how the results relate to the query. We explain those relations by highlighting how context, query and results are connected. When the user hovers over a query term (named entity), we highlight the terms in the focus paragraph, from where this named entity has been extracted. We also highlight the query term itself in the same color in order to make the connection obvious. In a similar fashion, we also highlight the corresponding result surrogates in which this query term appears, explicating the relation between query and results.

As both relations (context - query, query - result) are bi-directional, the visual explanation can also be initiated by hovering over a result. When hovering over a result, the corresponding query terms are highlighted alongside with their origin in the paragraph. As these query terms may be related to further results, those results are also highlighted accordingly. By the latter, we also display the relation among results (via shared query terms). An example of visualizing this triadic relationship is illustrated in figure 8.1 on the following page

(screenshot of the prototype used for data collection in the explanation group, cf. section 4.1.2.3 on page 32).

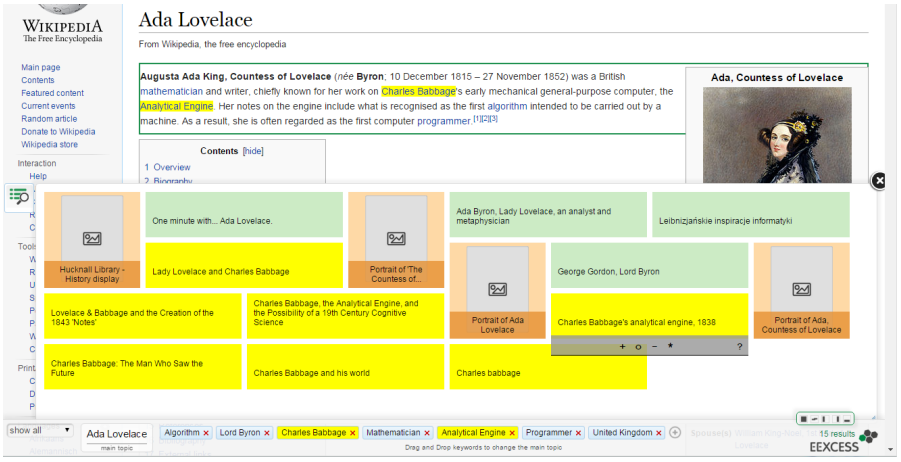


Figure 8.1: Screenshot of the prototype with explanation feature used for data collection. The mouse is positioned over the result "Charles Babbage's analytical engine, 1838". Corresponding query terms "Charles Babbage" and "Analytical Engine" are highlighted along with their origin in the paragraph. Further results related to those query terms are also highlighted.

Our visual explanations explicate from where in the paragraph query terms have been extracted, not how (named entity recognition). Similar for the results, where the ranking of the retrieval is a black box even to our system. Hence, visualizing the triadic relationship does not increase actual transparency, but potentially perceived transparency.

Regarding the influence of explanations, we had the following three hypotheses:

- **H1** Explanations reduce the amount of neutral ratings (*effectiveness*)
- **H2** Explanations increase the amount of positive ratings for automatic queries (*transparency / effectiveness*)
- **H3** Explanations reduce the time to arrive at a decision (*efficiency*)

The first hypothesis relates to the *effectiveness*, i.e., helping users make good decisions. Our aim here is helping users make decisions *at all*. Results are rated neutral in our dataset, if users could not judge whether they are related to the context. By explicating the relatedness through the visual explanations, we aimed to simplify the judgeability.

Hypothesis two aims at increasing transparency and effectiveness. We assume that (previously hidden) relations between context, query and results are more obvious to the user.

The last hypothesis relates to efficiency: We assume that our visual explanations allow faster judgments.

8.4 RESULTS AND DISCUSSION

We evaluated the visual explanations based on A/B testing with the data collected for part ii, described in section 4.1 on page 29. In the cleansed dataset, 38 users had the explanation feature, 31 users did not have the explanation feature. The results for both groups are shown in table 8.1 As can be seen from the table, there was no difference in

Table 8.1: Results for explanation/no explanation group.

	rating				**proportion**		
group	total	-	o	+	$\frac{o}{\text{total}}$	$\frac{+}{\text{total}}$	timeout
explanation	1805	970	314	521	0.17	0.29	0.57
no explanation	1450	785	240	425	0.17	0.29	0.62

the ratio of neutral ratings ("cannot judge") to the total amount of ratings between the two user groups. Therefore we have to refute the first hypothesis **H1** as our explanations are not reducing the amount of results user cannot provide a rating for.

The same applies to the second hypothesis **H2**: the ratio of positive ratings to the total amount of ratings is the same in both groups. Our visual explanation does not increase the proportion of positively rated results.

The last column of the table shows the proportion of how often users ended the task because the time was over. Other possible options were full satisfaction with the retrieved results or deeming the search engine not being able to deliver results at all. For some tasks we had no indication, why users ended the task. We count these as timeouts also. Ignoring those tasks where the indication is missing, the difference is more pronounced (0.48 vs. 0.58). Therefore, we can confirm hypothesis three **H3**, our implementation of a visual explanation increases efficiency, allowing users to arrive at decisions faster. Unfortunately we did not collect individual timings per user. Hence we cannot quantify the difference in efficiency but only observe its existence.

While the evaluation confirms hypothesis **H3**, the first two hypothesis about effectiveness are rejected. A potential reason for the latter is that visual explanations do not change the ground truth: We asked participants to rate neutral, if they cannot judge the relatedness, e.g. due to lack of domain knowledge. The visual explanations do not alter the participants' domain knowledge, hence in that case, they are not able to judge the relatedness with visual explanations either. Hypothesizing that visual explanations decrease the ratio of neutral ratings (or increase the ratio of positive ratings), we assumed participants rated results they could have identified as related upon close

inspection as neutral. It seems they rated very precisely, both with and without visual explanations, which is also supported by the high ratio of timeouts.

We conclude that visual explanations are beneficial for efficiency. The ratio of timeouts decreases with visual explanations, which confirms our third hypothesis of participants being able to arrive at decisions faster.

9 QUERY HISTORY VISUALIZATION

A common phenomenon in Web search is that users re-access Web resources that have been found in the past. Accessing previously found information is different from information seeking, e.g., by being more targeted and more directed involving recognition and recall activities [34]. While active strategies (i.e., explicit storage of the information) would support information re-finding, passive strategies with no explicit storage are much more common, especially when search tasks are interrupted [134]. Such passive strategies require to recall how or where the information was found previously. The difficulty of recalling where and how information on the Web was accessed is known as the "Lost in Hyperspace syndrome" [39]. This difficulty of recalling information is even more prevalent in zero-effort queries, as the search is not actively triggered by the user, but performed implicitly.

While strictly speaking the "Lost in Hyperspace syndrome" refers to the navigation of hypermedia only, an analysis of human information seeking models shows a similar behavior in the context of Web search. Models of human information seeking describe and structure the way humans search for information in an information source (for an overview see [71]). These models define human information seeking as an iterative process in which query reformulation is a common step (e.g., [13, 126]). Usually, multiple steps have to be taken and multiple query reformulations are necessary before the information need is fully satisfied. The demand to include a search history for supporting the query reformulation stage has been explicitly stated (e.g., [71, 182]). A search history also supports information re-finding for interrupted search sessions, which have been found to occur in 40% of all information seeking tasks in the study of Sellen et al. [178]. Resuming a search from a previous query relying on human memory has been shown to be only accurate in 72% of the time [201]. In the information retrieval community, automatic query refinement is an active research topic (e.g. [86]). This indicates that, in general, queries posed by users are underspecified and need to be refined iteratively.

9.1 OVERVIEW

We propose QueryCrumbs, a simple-to-understand, compact visualization for accessing, altering, and resubmitting previously issued queries. The concept is similar to bread crumbing interfaces as navigational aid for websites [114]. Figure 9.1 on the following page shows

the conceptual idea of the QueryCrumbs visualization. Each query

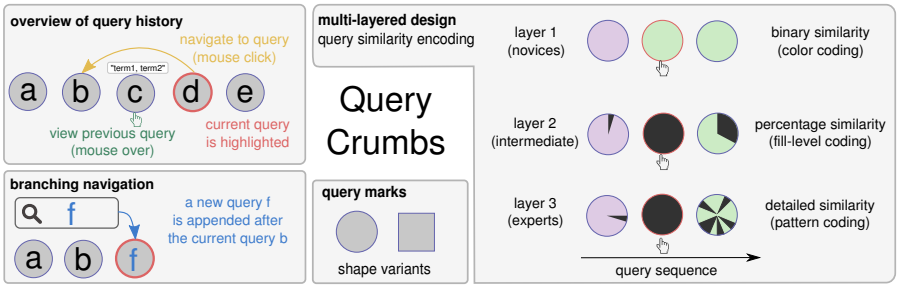


Figure 9.1: QueryCrumbs visualization concept. Previous queries are shown. Navigating back to a previous query reissues the query (top left). Issuing a new query from a previous one removes previously subsequent queries showing only the current path of interest (bottom left). Two different shape variants (bottom middle). Query similarity is based on the similarity of the search result lists and can be encoded with different levels of detail (right).

is represented by a mark, the position of the mark indicates the position of the query in the sequence of queries and different notions of query similarity are encoded in the mark's visual attributes. We introduce three different measures for query similarity to capture the general relationship between queries and corresponding mappings to the marks' visual attributes. The similarity is measured on different levels of detail, suitable for different user groups and tasks. In order to evaluate the usefulness of this visual representation, we pursue a layered interface design approach [181] introducing different notions of similarity in each layer. We evaluate the visualization and interaction design in a formative user study with novices. Additionally, we perform a think aloud test with experts in information retrieval to investigate which conclusion can be drawn about the search engine when using the visualization with the advanced similarity encoding. Finally, we evaluate the actual usage of QueryCrumbs outside the lab environment with software logging in a long-term study. Concretely, we...:

- ... introduce a human querying model as the conceptual basis for search history visualizations.
- ... propose and evaluate QueryCrumbs, a search-engine agnostic, compact and interactive visualization supporting overview and navigation of the query history while taking up minimal screen space (i.e., for mobile environments or with minimal impact on current search result page designs).
- ... account for universal usability by applying the multi-layered user interface design method to the design of the visualization.
- ... show, that search experts can gain valuable insights into search engine internals and which conclusions they can draw.
- ... demonstrate uptake in a long-term study.

The remainder of this chapter is organized as follows: After discussing related work, we describe the human querying model in section 9.3 on page 114 and from that derive the conceptual idea for the visualization in section 9.4 on page 116. Then, the multi-layered approach to visualization and interaction design is explained in detail in section 9.5 on page 118. The formative user evaluation described in section 9.6 on page 121 assesses the usability for the two layers designed for first-time and intermediate users. In a study with experts we then assess which conclusions can be drawn about the underlying search engine with the help of the visualization (cf. section 9.7 on page 130). We evaluate the actual usage of the visualization in section 9.8 on page 133 and conclude the chapter with a discussion of limitations and a summary.

9.2 RELATED WORK

We review insights on human querying behavior gained from web logs and human search models to motivate the human querying model as the conceptual basis for QueryCrumbs. Further, an overview of and design guidelines for search history visualizations, and the relationship to information re-finding behavior and related tools are presented.

9.2.1 *Human Querying Behavior*

Web query log analyses provide statistical data about human querying behavior. Broder [26] derived a basic categorization of web queries from analyzing AltaVista logs and survey data distinguishing transactional, navigational, and informational queries. Approximately 50% of queries are informational queries, with which users seek a certain piece of information. The author observed that especially informational queries range from a very broad description to a very narrow description of the information need. A similar observation was made for mobile search [8]. The authors classify queries on a continuum ranging from very specific to very general and 44% of the queries to be imprecise or general. 24% of the queries were classified as difficult because some information to construct a good query was missing, e.g., the name of an actor when searching for movies with this actor.

A detailed analysis on search sessions in AltaVista query logs is provided in Jansen et al. [82]. In this dataset 52% of users modified their queries. 48% of search sessions were single-query sessions, 32% of the sessions contained three or more queries. In another study on the same dataset 37% of all queries were found to be query modifications of various types [81]. A taxonomy of query reformulation strategies is given in [75], including stemming and addition of words. The authors built a classifier to automatically detect reformulations

and conclude that most of the reformulations can be accurately detected (90% accuracy). They note that some modifications are nearly impossible to detect automatically (e.g., the rephrasing of the query "how to calculate nutritional values" to "weight watchers calculator").

These statistics indicate that human querying is an iterative process in which query refinement is a common step, a result that is reflected in models of information seeking behavior reviewed in the next section. The results from the discussed log analyses are reflected in the human querying model we present as basis for QueryCrumbs.

9.2.1.1 *Search Experts*

In terms of query behavior, search expertise has typically been investigated along the following dimensions [215]: query attributes (choice of search terms, query length and syntax) and search strategies and tactics. Aula et al. [7] report the use of widely different definitions of what constitutes an expert, ranging from more than five hours of browsing a week to at least three years of extensive professional experience. For log-based studies, the use of advanced query syntax (such as quoted queries) has been used to identify advanced searchers [216]. Bates has formulated a set of potentially useful search tactics [12], which have later been refined and extended by Smith [187]. We follow the notion of related work in defining experts as being capable to use advanced query operators and applying a variety of different search tactics. In particular, their search tasks are usually more complex, sense-making tasks and therefore, an explorer-type behavior [214] with longer sessions, containing more than the average two queries [90] is expected. This increased complexity is reflected in QueryCrumbs by a multi-layer approach, providing more details to advanced users.

9.2.2 *Human Search Models and Search User Interfaces*

Multiple models for the human information seeking process have been proposed. The model of Shneiderman et al. consists of the four-stages *formulation* (generating the query), *action* (starting the search), *review* of results, and *reformulation* [182]. Similar models were later proposed by Sutcliffe & Ennis [193], and Marchionini & White [126]. All these models describe an iterative process and include the need of query reformulation and potential backtracking.

In the "berry picking" model of information seeking [13], the information need is not static, but changes with the resources found with each (reformulated) query. Starting with an initial query humans evaluate the results, which leads to new thoughts and to a rephrasing of the query. By repeating this process, the user discovers new resources and thoughts, which is likely accompanied by query modifications.

Mulhem & Nigay [136] applied Normans' seven stages of action model [139] to information retrieval tasks. Specifically, they identify two user-system distances spanning the gulf of execution. The first distance is the input semantic distance, which describes the challenge of identifying the semantic description of the information need (e.g., the right concepts for the specific search engine). The second distance is the input articularity distance, which describes the challenge of identifying the physical description of the search query (e.g., the right keywords and search operators). Both distances describe gaps between the conceptual model of the user and the system that cannot be bridged in general. Reasons include the dynamic nature of information sources, concept drift in systems and users [55], and that in some usage scenarios the search engine is a black-box from the view of the search user interface. These two distances point towards the necessity for query adaptation and rephrasing.

The analysis of the models shows that query reformulation, adaptation, and backtracking need to be supported in a search history visualization, which is reflected in the QueryCrumbs interaction design.

9.2.3 *Search History Visualizations*

While the above mentioned models implicitly indicate the requirement for user interfaces supporting search history navigation, this need has been explicitly stated by multiple authors (e.g., [71, 182]). A commercial example is Google's Wonderwheel, a visual tool for interactively finding related queries [190]. A query is represented as a node, clicking expands the node and shows related queries. Wonderwheel allows arbitrary branching from nodes and leads to a complex graph structure. It supports navigation in the query space, users can go back to previous queries and get new queries suggested. Wonderwheel is designed to focus on exploration of the information space (leading to the complex graph structure), while QueryCrumbs focuses on exploitation of the information space.

Komlodi et al. present design guidelines and examples for search history visualization based on a study with librarians [100, 101]. This work is similar to ours, while their target user group is different (search experts vs. casual searchers). Their interface follows the information webspace concept [35], and therefore has richer interactions and is much more complex. Conceptually similar to our work are bread crumbing interfaces [114, p. 221f] introduced as navigational aid for websites. In bread crumbing interfaces the navigation history is presented in a compact manner, but similarity of visited sites is not shown.

The notion of flow has been adapted to user interface design [14]. One aspect to support flow is to make the current state in the user

interface transparent, alleviating the problem of very limited human short term memory. In the context of this work, this can be translated to presenting the current state of information search in context to previous searches.

9.2.4 *Information Re-finding*

While also relying on history mechanisms, information re-finding differs from information seeking [34]. Information re-finding tasks can be categorized into short-term (retrieving just visited information), mid-term and long-term (re-finding information after months or years) tasks [44]. Re-finding behavior was also observed, when an information seeking task is interrupted [134] and is not well supported by standard Web browsing interfaces [213]. A study by the same authors showed that while being interrupted 58% of users did nothing to explicitly store the retrieved information (passive storage) and relied either on passive (memory, open browser windows) or active retrieval mechanisms (re-querying or browser history) [134]. Similar findings have been made in a different study, where 66% of users do nothing to save search results, but re-assess the page via search and 54% of participants reported to use this method at least once a week [27]. A study on query reissuing revealed that 28% of the time queries are misremembered when being asked for it one hour later [201].

Tools supporting an active strategy for information re-finding are for example Session Highlights [83] and a Firefox plugin for storing Web page summaries [46]. While these tools require an explicit user interaction to store the information, the SearchBar [134] and SearchPad [19] assume a passive user behavior for information storage. Both, SearchBar and SearchPad are centered around search queries, store additional documents and context and require a complex management. Re-finding tools assuming passive user behavior have also been proposed for other application areas, such as history of visualizations [72], or information re-finding within a Web page [45]. SearchBar and SearchPad are the most similar tools to QueryCrumbs, but require much more screen space and complex information management.

In summary, our approach is search-centric, and covers short-term to long-term re-finding strategies for users that pursue a passive information keeping behavior.

9.3 HUMAN QUERYING MODEL

Before introducing the concept for the QueryCrumbs visualization, we define the underlying human querying model. Human information seeking models capture the process required to satisfy a user's information need, they do not model the querying process explicitly. Deriving the information need from a query or a set of queries is

ongoing work in the information retrieval community [86]. Multiple queries might reflect the same information need and different information needs might be expressed by the same query. An example for the former are the two queries "buy mobile phone" and "buy phone", an example for the latter is the query "java" where a user might seek information for the island, the coffee, or the programming language.

As the queries and the retrieved results are the only data that is generally available to a search client, we introduce a human querying model on the basis of this data. This makes the querying model search-engine agnostic, i.e., we do not make any assumption about the type, nature or amount of the backend search systems. The model has the form of a graph, in which nodes correspond to queries and edges reflect query modifications (see figure 9.2). A user starts with

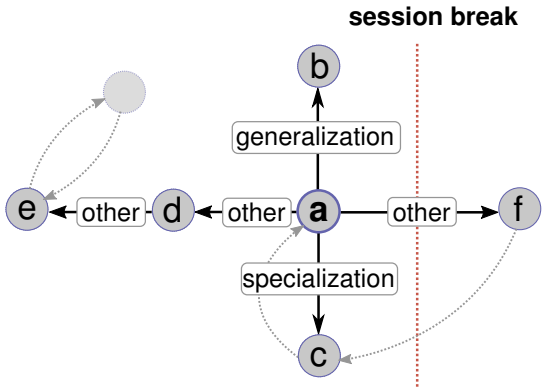


Figure 9.2: Human querying model. Query modifications with the same search intent include specialization and generalization. A session break occurs when the search intent changes.

an arbitrary query *a*. When the results for this query do not satisfy the user's information need, the user can either generalize the query (if it was too specific) leading to query *b*, specialize it (if it was too generic) leading to query *c*, or modify it in other ways leading to query *d*. Other modifications capturing the same search intent include the use of synonyms or rephrasing. When the search intent changes with the modification of the query (*f*), a session break occurs. Figure 9.2 only captures the trellis of the underlying graph, subsequent query modifications could lead to cycles (as indicated by the light gray node in the figure). The general graph capturing all queries and their relations has a infinite number of nodes (because there is an infinite number of potential queries). Users navigate through this general graph, and the queries a user issues correspond to a (potentially cyclic) sub-graph.

We illustrate this model by an example. Imagine the user task "prepare something for a children's birthday party tomorrow". The derived information need might be "find an easy to prepare recipe for cookies". Query *a* could be "baking cookies". A generalization of this query would be "cookies" (query *b*), a specialization "baking cookies

gluten-free" (query *c*). A modification with the same search intent could lead to the query "easy cookie recipe" (query *d*) or subsequently to the query "gluten-free baking" (query *e*). We note that the query modifications cannot necessarily be classified exactly by only knowing the query string. For instance, the new query "baking muffins" might indicate a session break or a query modification in the session with the same search intent of "preparing something for a children's birthday party". However we argue that for the purpose of query history visualization an exact classification is not necessary and therefore we kept the human querying model as simple as possible.

This human querying model can be seen as a special case of an information seeking model. It does not make any assumption about the underlying information need or the search goal, but captures the querying and query modification process. This simplification allows to approach the visualization of the human search process and is used as a basis for the concept of the QueryCrumbs history visualization.

9.4 QUERYCRUMBS CONCEPT

Conceptually, QueryCrumbs visualizes the most recent path through the general querying graph, i.e., the user's history of search queries, supporting the 5 users tasks:

- **Overview:** Get an overall overview of the query history, i.e., the sequence of queries.
- **Navigation:** Navigate back to previous queries, thus be able to easily access results from previous queries.
- **Simple comparison:** Identify similar searches conducted in the past, and thereby identify search sessions and session breaks.
- **Quantitative comparison:** Compare the quantity of overlapping search results for different queries. Investigate how the result set changed quantitatively.
- **Qualitative comparison:** Compare the search result ranking across queries. Investigate how the result set changed qualitatively, i.e., identify the location of result set changes.

Figure 9.1 on page 110 shows the concept of the visualization and interaction design. In the previous section we introduced the human querying model as a general graph. For the introduced tasks it is not necessary to show the graph, thus we present only the user's navigation path and unroll any cycles (i.e., display each graph node as often as it has been visited). This choice is further discussed in section 9.9 on page 138. Query marks are arranged from left (older) to right (most recent) to give an **overview** of recent searches. We propose a simple mouse-over interaction for previewing a previous query (i.e., show the query terms for this query), and a mouse click for **navigating** to a query. Navigation to a query means reissuing this

query. In a first design we constantly displayed the query terms for all queries. A preliminary study showed that this much text is (i) more hindering than helpful for users and (ii) poses a layout problem for long queries, which cannot be solved in limited space for arbitrary query lengths. Thus, in the subsequent design we only show the query terms for the current query on mouse-over. The navigation interaction corresponds to Shneiderman et al.'s guideline for supporting the query reformulation stage: allow users to review, alter, and resubmit queries [182].

9.4.1 *Layered Approach*

Intended for the use with general search engines and thus casual users, the visualization should be understandable without instructions by novices and first-time users. To this extend, we apply a multi-layer interface approach [181] to the visualization design. The visualization has three layers depicted in figure 9.3. Layer 1 is designed for the tasks

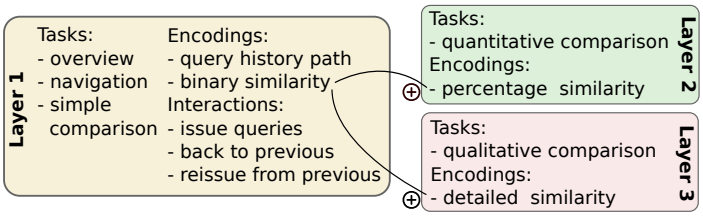


Figure 9.3: Multi-layer visualization concept showing tasks, encoded data and interactions for all layers. Layer 2 and 3 extend the similarity notion of layer 1.

"overview", "navigation" and "simple comparison", and therefore introduces all interactions. Layer 2 and 3 add the more complex notions of similarity, and are designed for the tasks "quantitative comparison" and "qualitative comparison", respectively. Although it might seem that layer 2 and 3 only add minor information, we found in preliminary studies that the different similarity notions are hard to understand for users, and therefore we transferred them to additional layers.

In the design we also considered potential adaptation for mobile devices. Adaptability is ensured by (i) compactness of the visualization and (ii) simple interactions that can be performed with either a mouse, or on a touch display.

9.4.2 *Measures for Query Similarity*

The three comparison tasks introduced at the beginning of this section require a notion of similarity between queries.

Query similarity can be either calculated on the basis of the query string or on the basis of the results returned. Because the former does

not capture semantic similarity, (e.g., the terms "car" and "automobile" are considered as different), we focus on query similarity based on the retrieved results. Even though the two queries"car" and "automobile" are syntactically different, they could lead to similar results when posed to a search engine. Thus, deriving similarity based on result sets can capture semantic similarity and renders the visualization search-engine agnostic. In the following we introduce three different measures for query similarity capturing different levels of detail.

Typically, search engines return a ranked list of results for a query k. Let this ranked list be denoted by $R_k = [r_k^1, \ldots, r_k^i, \ldots, r_k^n]$, where r_k^i is the $i-$th result for query k. Because users of Web search engines only access the top items in the result list [87, 149], the similarity calculation is based on the top τ items, yielding the ranked list R_k^τ. The ranked lists can be directly used to assess the **qualitative comparison**, i.e., comparing which elements in two result lists are similar when viewed side-by-side.

Two queries can be compared pairwise based on their result list and the overlapping elements can be identified. With this similarity, it can be judged if a result that is present in the list R_i is also present at any position in the list R_j. Let $L_k^\tau = \{r_k^1, \ldots, r_k^i, \ldots, r_k^\tau\}$ be the (unordered) set of results. The similarity sim_r of two queries can then be calculated as the Jaccard coefficient [197] on the two result sets (as opposed to using the ranked lists for the detailed similarity).

$$sim_r = \frac{|L_i^\tau \cap L_j^\tau|}{|L_i^\tau \cup L_j^\tau|} \in [0,1] \tag{9.1}$$

sim_r can be expressed as a percentage to which we further refer to as **percentage similarity**. This similarity corresponds to the user task **quantitative comparison**.

A binary indicator variable s_r can be obtained by introducing a similarity threshold $\theta \in [0,1]$, and is calculated as follows:

$$s_r = \begin{cases} 1, & \text{if } sim_r \geq \theta \\ 0, & \text{otherwise} \end{cases} \tag{9.2}$$

We further refer to s_r as **binary similarity**. This similarity corresponds to the user task **simple comparison**.

Figure 9.3 on the preceding page shows an overview of the presented data, similarity notions and user interactions available in each layer.

9.5 QUERYCRUMBS VISUALIZATION

The concept of the visualization described in the previous section is implemented in D3.js [24] and released[1] under the MIT license.

1 https://github.com/EEXCESS/QueryCrumbs features the source code and a demo video for layer 3 (expert users)

9.5.1 *Visualization and Interaction Design*

The basic design of the visualization incorporates a mark for each query. Query similarity is encoded in the mark's visual attributes and position is used to show the query sequence (time). Figure 9.4 shows an example of the QueryCrumbs visualization for all three layers with either circles or squares with fixed size as visual marks for a single query. The currently selected query is outlined with a red border.

(a) Layer 1 - novice users

(b) Layer 2 - intermediate users

(c) Layer 3 - expert users

Figure 9.4: QueryCrumbs layers for two different visual marks (left: circle, right: square) for queries "ada", "ada lovelace", "ada byron", "ada language", "ada programming", and "alan turing". Current query is highlighted in red.

In layer 1 (see figure 9.4a) the simple similarity s_r from equation (9.2) on the preceding page is encoded by color. Similar queries have the same color. We used a color map for qualitative data from ColorBrewer [25]. In a sequence of queries a new query q might be similar to more than one previous query a and b, but a and b might not necessarily be similar to each other. All choices to resolve this coloring ambiguity significantly increase the perceptual complexity of the visualization. We chose to avoid such a complexity by choosing the color of the most recent, similar query instead. This coloring scheme tends to (i) color the new query with the color of the current session if it belongs to it, and (ii) visually shows if a same query or session was issued in the past (with a different session in between).

In layer 2 (see figure 9.4b), the percentage similarity from equation (9.1) on the preceding page is additionally encoded in the fill-level of the mark. For circles, the angle of the filling and for squares the height of the filling corresponds to the percentage similarity. In the user evaluation we also addressed the question with which form (cir-

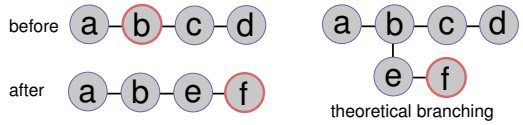
cles or squares) the similarity can be more accurately interpreted by users.

Layer 3 (see figure 9.4c on the preceding page) is designed for experts to assess the differences in two search result rankings in more detail. The query mark is divided into τ equally-sized sub-marks, one for each element in the result list, similar to Dense Pixel Displays [93], in which a single data item corresponds to one pixel. The sequence of sub-marks encodes the rank in the search result list and corresponds to the Western reading direction for squared marks and to a clockwise reading for circles. Queries are compared pairwise, the hovered query (i.e., the query under the mouse pointer) is compared to all other queries (and itself). If a result from the hovered query is present in the result list of another query, the corresponding sub-marks in the hovered and the other query are colored dark gray. In a first version of QueryCrumbs the sub-marks of the current query were only colored dark if they reappeared in another result list, different from the hovered one. However, preliminary user studies showed that this was confusing to interpret for users. Users did not understand why results that are currently displayed in the accompanying search result list are not marked in the query mark. Therefore, we decided to compare the hovered query also to itself, which colors the sub-marks for all results in the list dark gray. This comparison to itself further provides an indicator of how many results have been retrieved (if less results than available sub-marks have been retrieved).

9.5.2 *Navigating the History*

The QueryCrumbs visualization has two simple interactions for navigating the history. Mouse over allows to access basic information about a previous query. In layer 1 this information is the query string, in layer 2 and 3 additionally the similarity to other queries is visualized. An example is shown in figure 9.4 on the previous page. A mouse click highlights the selected query mark and reissues the query. If the user issues a new query (being on a previous-to-last query) this would mean a branching of the query history as shown in figure 9.5a on the facing page.

Because this branching could get rather complex as outlined in the introduced querying model, we remove all the query marks on the right of the current query (more recent queries) and append a new query mark. To make the change in the visualization better perceivable, the transition in the layout is animated. Figure 9.5b on the facing page shows the step before and after the transition when the query “ada lovelace portrait” is issued from the second query in the history “ada lovelace”.



(a) Conceptual Branching

(b) Branching in the Visualization

Figure 9.5: Query reissuing and branching. (a) Concept. Initial history, query "b" is selected (top left). Theoretical query tree after two new queries "e" and "f" (top right). Visualized trellis of the tree (bottom left). (b) Example. QueryCrumbs before and after issuing a query from a previous one.

9.6 EVALUATION WITH NOVICES

In the user evaluation we wanted to assess whether the visualization can successfully be used (understanding the visualization and interactions), which benefits users see, and whether they would use it in the future. We posed the following hypotheses:

H1: Layer 1 can be understood and used successfully **without** instructions. This comprises the following visual encodings and interactions (cf. figure 9.3 on page 117):

- "issue query" (perform interaction, understanding of change in visualization)
- "back to previous"-navigation (perform interactions, understanding of change in visualization)
- "issue from previous" (understanding of change in visualization: subsequent queries removed, new appended)
- "similarity encoding by color" (understanding)

H2: The percentage similarity coding (layer 2) is understandable **with** instructions. There is a difference in using the two different marks as query representatives (squares or circles).

H3: When having experience with QueryCrumbs, users tend to use it in a real-world usage scenario.

For **H2** we expected a difference for the two different marks, because reading the percentage similarity from the fill-level in a circle and square requires interpretation of two different visual features which are know to have different acuities [212]. Interpretation in the human visual system relies on angle perception in the circle, and on area perception in the square.

9.6.1 *Design*

We used a between subjects design with the independent variable *form*, i.e., the type of query marks. The independent variable has two levels, it can either be a square or a circle as shown in figure 9.4 on page 119. Dependent variables are *completion time* (in seconds), *task success* (binary) and *understanding* (binary). *Completion time* was measured automatically in the test user interface accounting for network latencies. *Task success* measured the correctness of the performed interaction (whether the visualization is in the intended state after the interaction) and was judged by the evaluator. The variable *understanding* captures whether the user was able to interpret the state of the visualization and was assessed by questions users had to answer after performing a task. The correctness of the answer was then judged by the evaluator.

In the questionnaire we also asked for perceived beauty, perceived helpfulness, expected uptake, and which layer participants would prefer (preferred layer). Preferred layer has either value 1 (for layer 1, simple similarity coding) or value 2 (for layer 2, percentage similarity coding). All other variables were assessed using a five-point Likert scale, with "1" coding the worst value and "5" encoding the best value (e.g., 1 - "not helpful at all", 5 - "very helpful").

9.6.2 *Participants*

20 German-speaking volunteers (undergraduate and post-graduate students) participated in the 30 minutes evaluation, 10 males and 10 females. The age of the persons ranged between 20 and 33 years with 50% of the participants being between 22 and 26 years. All had normal or corrected-to-normal vision. One participant stated to be a novice computer user, 10 participants rated themselves as intermediate users and 9 as computer experts.

9.6.3 *Tasks*

Table 9.1 on the facing page gives an overview of the evaluation tasks and the measured variables task success (S), understanding (U), and completion time (T). Task set A (task T-A1 to T-A5) was performed with layer 1 of the visualization (see figure 9.4a on page 119). With task set B layer 2 with the fill-level encoding the percentage similarity (see figure 9.4b on page 119) was evaluated. Task T-C is designed to assess potential uptake of the visualization. T-C is a creative task, asking users to search for related material on a blog post they are writing. For this task users were not given explicit instructions on whether to use the QueryCrumbs visualization. They were only told to solve the task with the evaluation user interface, and were free to choose the visualization as an additional tool. We intentionally scheduled this

Table 9.1: Task overview (measured variables: S - success, U - understanding, T - time, I - number and type of interactions). Grayed variables were measured, but are not the focus in the evaluation. Tasks in italic font are used to prepare the visualization.

ID	Task Description	Measures			
		S	U	T	I
	Trial 1 - *LAYER 1*				
T-A1	*issue queries*	✓		✓	
T-A2	back to previous	✓	✓	✓	
T-A3	issue from previous	✓	✓	✓	
T-A4	*issue queries*	✓		✓	
T-A5	estimate binary similarity	✓	✓		
	Trial 2 - *LAYER 2*				
T-B1	estimate percentage similarity	✓	✓		
T-B2	*issue queries*	✓		✓	
T-B3	estimate percentage similarity	✓	✓	✓	
T-B4	estimate percentage similarity	✓	✓	✓	
Trial 3 - *OPTIONAL USAGE OF QUERY CRUMBS*					
T-C	writing a blog entry	✓			✓

task at the end of the evaluation, where users already had experienced what they can do with the visualization. For Task T-C we counted how many users used QueryCrumbs, and which interactions (I) they performed with the visualization.

The query issuing tasks T-A1,T-A4, and T-B2 required users to type a query in a search field and are used to prepare the visualization for the subsequent tasks. We report them, because we compare the completion time of these tasks to the tasks which required issuing queries using QueryCrumbs. Task T-A3 also required users to issue a query (with QueryCrumbs) and was used to measure understanding, i.e., whether users can correctly interpret how and why the visualization's state changed. The instructions for the query issuing tasks (T-A1, T-A3, T-A4, T-B2) were the following (translated from German): *Enter the search terms [...] in the search box.* For T-A3 the task instructions also contained the question *Please explain how the visualization has changed.* The instruction for the similarity estimation tasks were *Which of the previous queries are similar to each other?* and *Please estimate the similarity of the queries X, Y and Z to each other*. Users were asked to mark their

estimate, selecting one of the values 0%, 25%, 50%, 75%, 100% for all query pairs.

Task T-C was formulated as follows (translated from German, shortened): *You want to write a blog entry about the life of Ada Lovelace. You are looking for images to illustrate your blog entry. Use the browser extension to find relevant images, copy them to a text editor and provide a short description of the image content.* Note that in this task users were not explicitly asked to use the QueryCrumbs visualization, but the extension in general. However, in a previous task (T-B2) the search queries that had to be input were "lovelace", "ada lovelace", "ada countess", and "ada byron", which would have been a good starting point for a search. The task counted as successfully solved, if users found five images that were relevant for the task.

9.6.4 *Test Material*

For the evaluation we used a browser extension that provides a sidebar alongside each Web page [165]. This extension accesses the Europeana collection[2], the European aggregator for digital museum objects, and was modified to collect the evaluation measures.Figure 9.6 depicts the sidebar. Users can input a query in the search field (1) and search

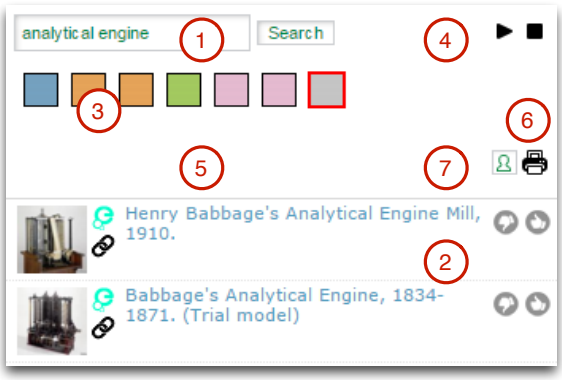


Figure 9.6: Evaluation user interface (result list cropped).

results are displayed in the result list (2) as document surrogates [126]. The QueryCrumbs visualization provides an overview of and access to previous queries (3). Users can start and stop an evaluation task using the controls on the top right (4). When the start button is clicked, an input field for the task id appears (5) and disappears after the task id was given. The correctness of the task id is ensured by the evaluator. The measures are stored in the browser's local storage and can be downloaded at the end of the evaluation (6). The layer of the visualization can be set in the user profile (7) by the evaluator.

The QueryCrumbs visualization was configured to show eleven previous queries. More queries were not required in the evaluation

2 http://europeana.eu – last accessed March 2020

and the size of the sidebar restricted the size of the displayable queries. The similarity calculations were based on the 16 top-most search results. The query similarity threshold θ was set to 0.1 for the binary similarity which was determined as a good threshold for visually indicating similarity in preliminary experiments.

9.6.5 *Procedure*

The evaluation comprised three trials. The first trial concerned layer 1 of the visualization (simple similarity and interactions) and was performed with task set A. In the second trial layer 2 (percentage similarity) with task set B was used. The third trial consisted of task T-C.

Before the first trial, participants obtained some general instructions on how to handle the evaluation interface. This contained the introduction of the start and stop task button and the instruction to start the task in the interface only after having read and understood the task instruction. For each participant the query history was set empty at the beginning, i.e., QueryCrumbs were not visible at the beginning. Because we wanted to evaluate whether the visualization in its basic design (layer 1) is understandable without explanations, participants did not receive any explanation about the visualization. Before the second trial, participants received a short introduction (one written paragraph) about the percentage similarity coding (layer 2). For each task in trial 1 and 2 we automatically collected the completion time. Because the results were retrieved online from a Web search engine, we controlled for network latency by subtracting the time it took the search engine to respond (which was below 1 second for each query). After the second trial participants had a short break and were told that from now on the completion time was not measured anymore. For the third trial participants received only the task instructions. At the end of the third trial participants filled out the post-study questionnaire.

Some tasks contained explicit questions (e.g., "Which of the search queries you issued are similar to each other?") and the participants had to speak out loud the answer. For other tasks the correctness of the answer could be judged by observing the state of the user interface. The experimenter noted the correctness for each task.

9.6.6 *Results*

We report the results of the formative user evaluation separately for each layer, the task measuring potential uptake and the questionnaire.

9.6.6.1 *Measured Performance for Layer 1*

Table 9.2 shows task success, understanding, and completion time for the tasks performed with layer 1. Values are aggregated over all users

Table 9.2: Results for layer 1 aggregated over all users. Showing mean and standard deviation for completion time (for task T-A1 and T-A2 there is one missing value). "n.a." indicates measure is not applicable for this task.

Task	**Success** [%]	**Understanding** [%]	**Time** [sec]
T-A1	100	n.a.	54 ± 17
T-A2	50.0	65.0	23 ± 22
T-A3	100	57.5	24 ± 18
T-A4	100	n.a	35 ± 13
T-A5	100	100	n.a.

independent of whether they used the circles or square condition. We found no influence of the variable *form* (circle or square) and thus omitted the values in the table. In task TA-1 and T-A4, in which users had to issue a query in the search field, we did not measure understanding. Similarly, for task T-A5 measuring completion time was not applicable, because users had to answer a question which required an explanation.

Nearly all, but Task T-A2 (back to previous), were successfully performed by all users. In Task T-A2 only 10 users (50%) successfully navigated back to a specific previous query. This means, the other 10 users either did not choose a previous query at all or did not choose the requested one. Conversely, the understanding rate for this task was high, 13 users (65%) still interpreted the state of the visualization correctly. This means, that although some users did not perform the interaction as intended, they still were able to understand the change in the visualization.

Task T-A3 (issue from previous) shows different results. Although all users successfully performed the interaction, only 58% could interpret the result correctly. This means, the navigation concept outlined in figure 9.5 on page 121 was understood by the majority, but not by all users.

If users successfully issued a previous query it took them 10 sec on average (Task T-A2, depending on task success). For a successful reissuing of a previous query users first needed to find the query in the visualization (mouse over), and then click the query mark. Typing a new query of similar length took 24 sec on average (Task T-A3). Interpreting the binary similarity of two queries (Task T-A5)

was successfully performed and also correctly understood (query representatives have same color) by all users.

*Summing up, we conclude that color coding of the simple similarity was well understood by all participants **without** instructions. Not all users (50%) performed the interaction for navigating back correctly, but 65% understood the interaction result. Reissuing a previous query is faster with QueryCrumbs than typing a new query.*

9.6.6.2 *Measured Performance for Layer 2*

Table 9.3 summarizes the results for layer 2. We do not report task

Table 9.3: Results for layer 2 aggregated over all users. Showing mean and standard deviation for completion time (2 missing values for task T-B3, one for task T-B4). "n.a." indicates measure is not applicable for this task.

Task	Understanding [%]		Time [sec]	
	Squares	Circles	Squares	Circles
T-B1	100	100	n.a.	n.a.
T-B2	n.a.	n.a.	35±7	45±27
T-B3	100	100	38±12	48±31
T-B4	100	100	24±6	21±10

success in this table. All tasks were executed correctly by all users, i.e., task success is 100% for all tasks. Also, all users correctly understood the encoding of the percentage similarity by fill level (understanding is 100%). There was no influence of the variable *form* (circles or squares) on the perception of the similarity coding.

It took users on average between 20 sec and 48 sec to complete a task. There was no significant effect of *form* on completion time for any task ($[F(1,18) = 0.201, p = 0.660]$ for task T-B2, $[F(1,16) = 0.945, p = 0.346]$ for task T-B3, $[F(1,17) = 0.737, p = 0.403]$ for task T-B4). This means the group using circles performed all tasks as fast as the group using squares, on average.

Summing up, we conclude that query result similarity was correctly interpreted by all participants in all conditions (100% task success), and the form of the mark had no influence on the completion time.

9.6.6.3 *Usage in Creative Task (T-C)*

In tasks T-C users were free to choose whether or not to use the QueryCrumbs visualization. Table 9.4 on the next page shows an overview over the usage of QueryCrumbs for this task. 12 participants (60%) used QueryCrumbs to find material for their blog post, 6 of them remembered and reissued a query that had been issued in a

Table 9.4: Usage statistics for Task T-C.

users using QC (any interaction)	12 (60%)
#users reissue with QC	10 (50%)
#users reissue with QC, from previous task	6 (30%)
total #queries	191
average #queries	9.55
#queries with QC	33 (17%)

previous task. The majority of those who used QueryCrumbs reissued a previous query (10 participants), 2 participants only used it for scrolling through the query history (mouse over). In total, 191 queries were issued in this task, 17% of the queries were reissued using the visualization.

In total, 90% of all users successfully completed this task, i.e., found five suitable images to include in the blog post. The task success rate was 91% for participants using QueryCrumbs, and 88% for those not using the visualization. Due to the limited amount of data no conclusions can be drawn for the influence of QueryCrumbs usage on completion time.

Summing up, we conclude that the majority of the participants chose to use QueryCrumbs issuing 1/3 of the queries with it.

9.6.6.4 *Questionnaire Results*

Table 9.5 on the facing page summarizes the quantitative values from the questionnaire. Generally users rated QueryCrumbs rather high in all categories, i.e., above the theoretical average of 3 for all variables. The similarity color coding and reissue interaction (both average rating of 4.1) were perceived as especially helpful. Users indicated that they would use both layers in the future (rating of 3.6 for both), but if given a choice, 15 (75%) would prefer the (feature-richer) layer 2. 11 users would prefer circles as marks and 9 squares. Only 5 users deviated in their preference from the condition they had been assigned to (e.g., had been working with circles and would prefer squares). This indicates a bias in favor of familiarity for this question.

When asked for comments for improvement, 19 participants commented on the overall user interface, and 11 participants commented on the visualization. Comments for the overall user interface included questions like "why is search re-executed and search results are not cached?" and "how do I close the extension?", and are not further investigated here. Suggestions for improvement of QueryCrumbs can be categorized into comments on "visual encoding", "interactions", and "alternative suggestions".

Table 9.5: Summary of questionnaire results. Showing mean and standard deviation (values from 5-point Likert scale, 1 - worst value, 5 - best value).

Question	**Rating**
beauty	3.8 ± 0.9
helpfulness of visualizations	3.4 ± 1.1
helpfulness color coding	4.1 ± 0.7
helpfulness fill-level coding	3.5 ± 1.2
helpfulness reissue interaction	4.1 ± 1.3
expected uptake layer 1	3.6 ± 0.9
expected uptake layer 2	3.6 ± 1.1
Choice	**Count**
prefer layer 1	5 of 20
prefer layer 2	15 of 20
prefer circles	11 of 20
prefer squares	9 of 20

For *visual encoding*, one user suggested a different color coding (remove gray as color), usage of gradients to make it more beautiful, or adding additional information to the marks (either the first letter of the query or showing the percentage value instead of the fill-level). Two participants would like to see the marks labeled (with the query terms), and two participants commented that there is no need for improvement ("thumbs up"). In terms of *interactions*, one participant suggested to add the possibility to delete queries from the history. Another participant would prefer to treat the query history as list in which no queries are automatically deleted when reissuing from a previous query. One participant suggested an *alternative* representation of the query history as a drop down list (similar to the browser page history).

Summing up, the questionnaire results show that users considered QueryCrumbs helpful and well-designed. Further we found high indication for potential uptake with preferences for layer 2.

9.6.7 *Discussion*

In the evaluation we distinguished between *task success* (successfully performing the interaction) and *understanding* (correctly interpreting the results). Results show that they are indeed not necessarily related. E.g., in task T-A2, users had only 50% task success on average for navigating back to a previous query, but still had understood the result

of the changes in the visualization (65% understanding rate). We would expect users that have understood the results, but not performed the task correctly to become more accurate when performing the same task again (not part of the evaluation). Similarly, for task T-A3 (issue a query from previous) the task success was 100%, but the understanding rate was lower (57.5%). In terms of Norman's seven stages of action model, for the interaction "back to previous" more users were lost in the gulf of execution, and for the interaction "issue from previous" more users were lost in the gulf of evaluation. This indicates areas for improvement, e.g., the speed of the transition could be decreased when a new query is issued from a previous one.

Thus, hypothesis **H1** can partly be confirmed. The visualization is usable and understandable without instructions (similarity coding, navigate back, issue from previous), but some users had problems navigating back to the requested query and interpreting the visualization state when a new query was issued from a previous one.

The similarity coding was understandable in both layers (hypothesis **H2**), and we found no influence of the form of the mark on accuracy or speed. We would have expected an influence of the variable *form* in layer 2 (fill-level coding), but the missing difference might be because we did not ask for the absolute percentage values. Users were only required to estimate the correct bin (of size 25%) which was feasible with both angle and area perception.

In the questionnaire users rated the helpfulness and beauty of QueryCrumbs high and in general stated that they would like to use it in the future (hypothesis **H3**). Most of the users (75%) would prefer to use layer 2 after having gained experience with both layers. The majority of users decided to work with QueryCrumbs in Task T-C, in which users were free to either do the task with or without QueryCrumbs. This is also an indication that users expect a benefit in usage and points towards future uptake. There was one participant who requested an improvement towards query history management, in this case the possibility to delete queries from the history. All other users seem to perceive QueryCrumbs as a support tool while searching (as intended) and do not think of it as a search history management tool.

9.7 EVALUATION WITH EXPERTS

The visualization was designed to show different levels of similarity of search result lists. Layer 3 that conveys the detailed similarity was designed for experts. The assumption is that the QueryCrumbs visualization can give information retrieval or search engine experts deeper insights into the querying process. The goal of the expert user study was to qualify potential insights experts can gain while interacting with the visualization during a search session. We further

wanted to understand their reasoning for the insights, and which patterns in the visualization indicate certain findings.

9.7.1 *Tasks*

Participants were given the search interface with the QueryCrumbs visualization and were asked to perform the following task first: "Input some queries and investigate the visualization. Please tell us, what you observe, what conclusions you draw and why you draw these conclusions." In the second task, participants received a set of prepared queries to input. In an optional third task participants could again use their own queries, if they expected more findings when trying to query again. The tasks took approximately 30 minutes to perform.

9.7.2 *Procedure and Participants*

We recruited eight experts with experience in information retrieval or search engines. Participants were either employed by the University or by a large German library. One participant was a graduate student, one a Post-Doc and six were PhD students. All participants were male, with an average age of 30, ranging from 25 to 37.

Participants were given an explanation of the visualization, with specific focus on layer 3 and had time to get familiar with the interactions. We chose the squared marks because we found no influence of a specific mark on performance in the previous evaluation (cf. section 9.6 on page 121). After that, the tasks and the pre-compiled query sets were introduced and participants were asked to perform the tasks while thinking aloud. A screencast (with audio) was taken during the experiment. The study ended with a questionnaire asking for potential usefulness and application areas of the visualization.

9.7.3 *Test Material*

We used the same search backend as in the previous evaluation (cf. section 9.6.4 on page 124) and slightly modified the user interface (similarity calculations based on the top-25 search results, buttons for task recording removed). We prepared 7 sets of queries that we assumed to lead to interesting insights. The sets of queries are listed in table 9.6 on the next page.

In set 1, both queries lead to the same result list, which means the search engine can be assumed to use stemming as a text pre-processing method. The queries in set 2 provide no overlap between the result lists, which indicates that synonyms might not be used by the search engine. The connection between the three painters (the artist group "Blue Rider") in set 3 is not made explicit in the search content. In Set 4 the second query leads to results that are a superset of the first

Table 9.6: Predefined query sets and QueryCrumbs for set 7.

1	car	cars	
2	car	automobile	
3	August Macke	Franz Marc	Paul Klee
4	August Macke	Macke	
5	haystack series	"haystack series"	
6	loom	loom weaving	
7	ada lovelace	"ada lovelace"	ada and lovelace
	ada or lovelace	ada AND lovelace	ada OR lovelace

query, with the specific property that the common results are at the beginning of the result list. Sets 5 and 6 have one single result in common between the two queries, in set 5 this is because the phrase query only returns a single result. Set 7 tests the implemented query language of the search engine, e.g., whether a list of terms is implicitly connected via the AND operator (which is true), and whether "and" and "or" in lower-case are interpreted as part of the query language or as query terms (the latter is true).

9.7.4 *Results and Discussion*

Table 9.7 on the facing page provides an overview of the search engine properties that experts have identified during the evaluation. All of the features expected when compiling the predefined result sets were identified, some by all expert users (e.g., stemming, synonyms, boolean query support). Most conclusions could be drawn about the query language: support for phrase queries, the syntax of the phrase queries and the default operator (AND) were identified. Additional insights were gained about reproducibility, i.e., the randomness of the ranking (2 users) and whether the proximity of terms in the documents has any influence on the final ranking (1 user). Own query sets were used to identify whether the query term order has any influence (2 users) and whether the search engine uses translation of query terms (1 user). Most users (7 of 8) performed the optional third task to test for further insights.

In the questionnaire, participants were asked about the usefulness, potential improvements and (further) application areas. Summarizing the comments, participants indicated that the QueryCrumbs interface is well suited for comparing search result lists. However, they remarked that result list comparison is not inherent to the search task

Table 9.7: Overview of search engine properties identified by experts using QueryCrumbs. Indicating the number of experts identifying the feature (column 2), and the predefined set or example queries used.

Category	#	Comment
	PRE-PROCESSING	
Stemming	8	Set 1
Lemmatization	3	*child* and *children*
Stopword Removal	3	*to be or not to be* and *to or not to*
Case-sensitivity	1	Insensitivity assumed, not tested
	ADVANCED NATURAL LANGUAGE PROCESSING	
Translation	2	*car* and *voiture*
Synonyms/Abbreviations	8	Set 2, *ww2* and *world war 2*
Concept Matching	1	Set 3
Named Entity Recognition	1	Set 4
	QUERY LANGUAGE	
Phrase Query Support	8	Set 5, Set 7
Boolean Query Support	8	Set 7
Boolean Operator Syntax	7	Set 7
Default Boolean Operator	4	Set 7, Set 6
	RANKING	
Reproducibility	2	1 user tested a query 8 times
Term Order	2	*star wars* and *wars star*
Query Term Proximity	1	Set 5

and performed only rarely. Accordingly, suggested further application areas comprised applications, where list comparison is a primary task, e.g., comparing friend lists in social networks. Regarding improvements, participants suggested to provide more details when comparing only two elements, e.g., by enlarging the elements or by a different visualization, such as a Venn-chart.

Summing up, the expert evaluation showed that insights about search engine internals can be acquired using QueryCrumbs.

9.8 USAGE EVALUATION

In the usage evaluation, we wanted to assess whether QueryCrumbs are actually used outside a lab study and if so, which features are commonly used.

9.8.1 *Test Material and Participants*

We implemented a browser extension for the Google Chrome browser to integrate with Google Scholar[3] (a search engine for scholarly literature). A screenshot of the evaluation prototype is depicted in figure 9.7. Google Scholar was chosen in order to target a larger audience as with

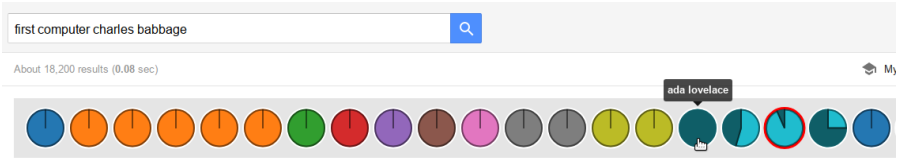


Figure 9.7: Screenshot of the evaluation prototype for usage evaluation (cropped). The current query is highlighted by the red circle (backtracked) and query under the mouse pointer compared to all others.

the cultural data backend used in the previous prototypes. While the largest audience might have been addressed by integrating QueryCrumbs into Google's general purpose search engine we still opted for Google Scholar, as it has a consistent interface. With Google's general purpose search engine, we faced frequent changes in the interface, both in terms of (CSS-)layout and identifiers. These changes would require frequent updates of our prototype and could invalidate evaluation results as the time between a change in the interface and the corresponding update of the prototype could have a negative influence. Further, with Google's general purpose search engine, several special cases need to be considered, as depending on the query, different verticals and/or info-boxes are displayed. Therefore we opted for Google Scholar, since the interface is simply the query box and a plain result list without frequent changes.

As visible in figure 9.7, the implementation included the first two layers, i.e., color and percentage similarity and used circle shapes for query marks. The similarity calculations were based on the first top-10 results, since that is the default returned by the search engine on the first page. The query similarity threshold θ was set to 0.1. That means, two queries are considered similar in terms of binary similarity (i.e., have same color), if they share a single result. Further, we followed a suggestion from the evaluation with novices and added the option to delete query marks with a double-click. During the evaluation we tracked the following interactions:

- **Draw**: Rendering of the visualization. Occurs every time a new query is issued.
- **Hover**: Shows the query terms and percentage similarity to all queries (comparison).

3 https://scholar.google.com – last accessed March 2020

- **Click**: Navigating back to a previous query and reissue that query.
- **Double-click**: Removes the query from the history.

Those interactions were logged over a time frame of 2.5 months. The interaction logs were sent to a lightweight node.js[4] server instance, storing them in a MySQL[5] database.

21 users volunteered to participate in the long-term evaluation, with the majority of them being computer science students or researchers. One user indicated, to not have used Google Scholar before. A globally unique identifier (GUID) was created for each user and interactions were logged anonymized with the user's corresponding GUID.

9.8.2 *Results and Discussion*

In this section, we present and discuss the evaluation results, first for the general usage of QueryCrumbs and then for the utilized features individually.

9.8.2.1 *General Usage*

Out of the 21 users who downloaded and installed the extension, 17 actually used QueryCrumbs at least once. To obtain statistics about the general usage, we calculated the usage duration as the amount of distinct days, the visualization was utilized at least once. This way, we get a more accurate measure for the usage than by simply taking the timespan between the first and last usage (which may contain several days without usage). The results are summarized in figure 9.8. 53% of the participants used the visualization for no

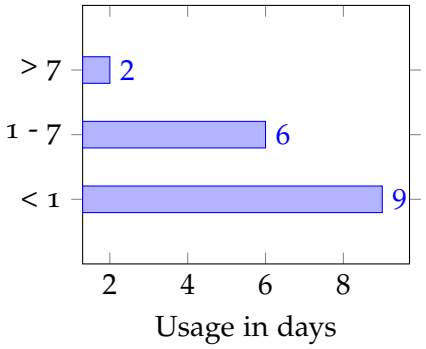


Figure 9.8: Results for general usage. Shows how long users that installed the extension used it.

longer than one day, 35% for less than a week and 12% for more than a week. The high number of participants using the visualization

4 https://nodejs.org – last accessed March 2020
5 https://www.mysql.com – last accessed March 2020

for only a week or less is partially explained by the selection of the search engine. Google Scholar's primary goal is the indexing of full text or metadata of scholarly literature. Even though the majority of participants were students, they do not use Google Scholar on a daily basis and we explicitly asked participants not to deviate from their regular behavior. Students typically use Google Scholar more frequently when writing their thesis or a seminar paper, but less frequently during a regular semester. Therefore, for this part of the participant group, using QueryCrumbs for at most seven distinct days is explained simply by them not using Google Scholar more often. Likely, participants who used the visualization at most for one day, visited Google Scholar only to try out the visualization on that day. They would not have visited otherwise and did not visit it afterwards. We assume, that the two users, who used QueryCrumbs for more than seven distinct days (28 and 35 days) were the researchers in the participant group, who also use Google Scholar on a regular basis.

Summing up, we see an uptake of QueryCrumbs on a scholarly search engine by participants, who use this search engine on a regular basis.

9.8.2.2 *Feature Usage*

In the second part of the evaluation, we investigated which features of the visualization are used and how often. Table 9.8 provides an overview of the feature usage. As can be seen in the table, adding

Table 9.8: Results for feature usage. Shows each feature, how many users used it and how often on average and in total.

Feature	Usage [%]	Average	Total
Add crumb	100	75	2733
Compare crumbs	100	177	3010
Navigate back	58.82	4	66
Delete crumb	35.29	5	88

a crumb and comparing queries was performed by all participants, while less participants utilized the navigation and deletion feature. Accordingly, the number of adding and comparing queries is way higher than the number of navigation and deletion actions performed. This behavior is expected, as a new crumb is added automatically every time a query is issued. Also, when performing a comparison, users quite likely take a look at several queries and compare them against each other. Another reason for the number of comparisons being higher than the number of added queries is that a comparison event can be triggered unintentionally. For example when moving the mouse pointer from the result list to the query input box, it

may be moved over a query crumb. Hence a comparison event is triggered, while the user did not actually compare queries. On the other hand, when the user compares queries simply by the color coding (binary similarity), we cannot detect this event via software logging and therefore have no information about how many queries were compared in that way.

The usage numbers for the navigation feature are naturally far lower than adding and comparing crumbs, as the navigation is used for backtracking in time. Typically, users do not repeat each and every query several times, but navigate back only to selected queries (if at all). Surprisingly, while the delete feature was used by less users than the navigation feature, they used it more often. Looking closer at this difference, in figure 9.9, we see that the median for the delete feature is even at zero. Whereas the statistics of the other features look

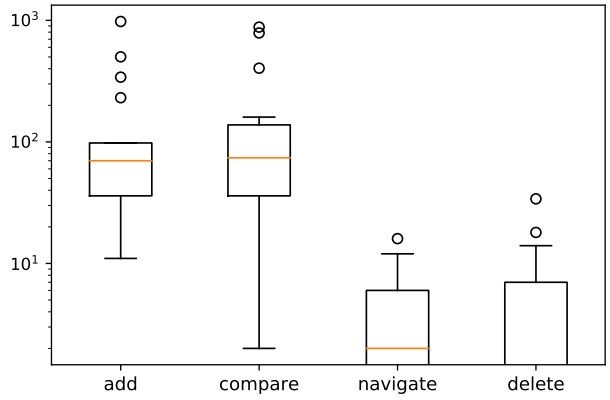


Figure 9.9: Boxplots for detailed statistics of the individual features (log-scale).

as expected, with a few outliers on the feature usage (those are the participants using the scholarly search engine on a regular basis). A potential explanation for not many participants deleting crumbs is that even though this feature was mentioned in the brief description of the browser extension upon install, participants may not have read that description or forgot about it. From the visualization itself, a double-click is not an obvious interaction to delete a crumb and therefore, participants might not have recognized that possibility. To counter that fact, a tooltip may be added in the future, if the user hovers over a crumb for a certain time, in order to inform about the delete feature from within the visualization.

Still, it is interesting to see that if participants were aware of the delete feature, they used it more often than the navigation feature. We assume that participants deleted queries without any relevant results in order to keep only relevant queries in their history. Removing irrelevant queries can be seen as a strategy between active (storing

relevant results) and passive (not storing anything). That intermediate strategy requires less effort from the user than actively storing relevant results. By QueryCrumbs logging the query history automatically, potentially relevant results are stored even for passive users. Whether a query returned no relevant results is easier to decide than whether a query returned (potentially - even if only in the future) relevant results, lowering the user's mental effort. The usage of this feature could also be used as an indicator for search engine evaluation. In search engine evaluation, typically clicks on a search result are seen as a positive signal. However, an abandoned query, i.e., a query where the user did not click on a result, does not need to be a negative signal. Li et al. [116] introduced "good abandonment", i.e., queries, where the user did not click on a result, but still could satisfy her information need. Such good abandonment occurs, when the search engine result page already satisfies the user's information need. The information need can be satisfied for example by answer boxes to factoid questions, or the result snippet might contain the required information. Hence, clicking a result is not necessary anymore. However, it is difficult to determine whether a user abandoned a query because she was satisfied or not. In such cases, the deletion feature could be used as an indicator for negative query abandonment.

Summing up, the visualization features were merely used as expected with the deletion feature having a higher usage than expected, indicating high value of that feature.

9.9 DISCUSSION

We choose to use a *simplification of a human querying model* for the visualization that does not show the explicit branching, but rather visualizes the history in a linear fashion. Human querying models [126, 134, 182] indicate that backtracking occurs mostly when users arrive in a dead end, i.e, modifying query terms does not lead to relevant results anymore. A recent study on Web search logs provides additional details on branching and backtracking behavior [50]. Because queries tend to get more complex at the end of a session, users backtrack to the more general query and start refining it. However, within one session (i.e., one information need) they hardly revisit a path they backtracked from. Also, removing branches after backtracking keeps the visualization small and comprehensible, while at the same time supporting the majority of query refinement steps within one query session. Explicitly displaying all query branches would result in a rather complex graph. Such graphs are hard to layout in a visually pleasing way and hard to navigate [74] and supporting small screens (e.g., mobile phones) would no longer be possible. Still, in the implementation of QueryCrumbs for the long-term study, we also store the backtracked branches without showing them. Storing the branches

enables potential future extensions, such as for example visualizing backtracked branches on demand for individual queries.

The query history visualization is limited by the space given by the user interface influencing the *scalability* as the number of queries grows. A search session contains 4 queries on average [50], while 67% of the sessions contain 1 or 2 queries, and 33% of the sessions contain 3 or more queries [82]. Even with the limited space for 11 marks (as in our evaluations with first-time to expert users) QueryCrumbs capture at least 2 search sessions on average. While more marks can be added, we estimate 2 search sessions as lower bound for a useful query navigation support and as a good trade-off between usefulness and support for limited screen-size. The implementation of QueryCrumbs for the long-term study allows for 26 query marks on a full hd display (at least 6 search sessions on average). Again, as we store the complete query history, future extensions are easily possible. Those comprise for instance scrolling back in the history or a user-defined setting of the query mark size (which implicitly defines the amount of displayable query marks).

Questionnaire results from the expert evaluation indicate that while QueryCrumbs can increase the transparency of the search engine by issuing sensibly selected queries, result list comparison is not performed by search experts on a daily basis. Therefore search experts suggested to extend the QueryCrumbs visualization to other application areas, in which list comparison is a primary task, e.g. comparing friend lists in a social network. When comparing result lists, search experts suggested to provide more details for the comparison, e.g., additional visualizations such as a Venn-chart. Both, search experts and lay persons suggested to reflect details of the comparison also in the result list, e.g., by highlighting results that also occurred in other queries. However, the result list is not part of QueryCrumbs, but specific to a particular search engine (interface). Therefore, such an extension has to be implemented specific to the search engine, whereas the QueryCrumbs interface itself is search engine agnostic.

The differences we observed in the uptake of QueryCrumbs in the long-term study are mainly governed by the usage of the selected search engine. The small amount of usage (in days) by the majority of users does not reflect a low usage of QueryCrumbs, but a low usage of the selected search engine. However, for the long-term study, we were limited by implementation/extension capabilities of common general purpose search engines as discussed in section 9.8.2.1 on page 135. We opted for the best tradeoff between targeting a large audience and a consistent user interface experience. Still, we observed an uptake of QueryCrumbs for those users that used the selected search engine on a regular basis.

9.10 SUMMARY AND CONCLUSION

We proposed QueryCrumbs, a simple-to-understand visualization for accessing, altering, and resubmitting previously issued queries. We applied a multi-layered interface approach to the design of the visualization and evaluated the layers intended for novices and intermediate users and experts.

The formative user study confirmed that the first two layers were well understood and usable with minimal to no instructions, and pointed towards which parts of the design could still be improved. A long-term deployment study with software-logging incorporating the first two layers of the visualization indicated actual uptake also outside the lab environment. The expert evaluation revealed that using QueryCrumbs and sensibly selected query sets, experts can gain interesting insights in the behavior of the search engine, for instance about the applied pre-processing and the ranking of results.

Part IV

REPRESENTATION OF RESOURCES

RQ 3

How to represent resources to facilitate keyword-based search?

10 INTRODUCTION

Regular Web-search is typically full-text search across documents, whereas search in digital libraries is based on object metadata. In particular cultural artifacts, such as paintings, archaeological finds, etc. are not represented by documents. However, this distinction is not strict, but there is an overlap between the two retrieval types in digital libraries. Our goal in this part of the thesis is to find a unified feature representation of resources, regardless the information source (text or metadata), in order to bridge the gap between full-text and metadata based retrieval.

Instead of laborious hand-engineering, those feature representations can be learned by solving an optimization problem [16]. We focus on unsupervised representation learning, i.e., learning task-independent features by defining an objective independent of the downstream prediction task. The flexibility in the definition of the objective in the unsupervised setting allows to define computationally efficient feature learning mechanisms and the representations to be used across several tasks. A key aspect in feature learning is to reduce the dimensionality of the original feature space in a way, that the relevant information is still retained, while noise is eliminated. Typically, methods that account for special properties of the data perform better than general dimensionality reduction methods, such as for example PCA [88].

10.1 OVERVIEW

In the next section 10.2 on the next page, we provide background information on the skip-gram model with negative sampling (SGNS), a representational learning approach presented in the context of natural language processing. Originally, the SGNS model was introduced to learn feature representations for words, but has later been extended to also learn representations for paragraphs or whole documents.

We combine different optimization techniques in order to decrease the convergence time (in terms of training iterations) of the SGNS model in chapter 11 on page 147.

We then discuss the extension of the SGNS model to networks via random walks, i.e., learning feature representations of nodes in a network in chapter 12 on page 159. In that chapter, we also investigate several extensions to the pure random walk strategy and complement them with our own. In particular, we investigate means to control the random walk behavior and abstraction levels of the original network.

Finally, we combine those models into a joint model for text and network information that can cope with the (partial) absence of one modality in chapter 13 on page 175.

10.2 BACKGROUND: SKIP-GRAM WITH NEGATIVE SAMPLING

This section describes the "skip-gram with negative sampling" model as introduced by Mikolov et al. [133]. It provides the basis for the models presented in the upcoming chapters. Readers who are familiar with the model may skip this section. The model became famous under the term "Word2Vec", named after the original c-implementation. Strictly speaking, Word2Vec includes two model-variants: Continuous Bag of Words (CBOW) and Skip-Gram with Negative Sampling (SGNS) [132]. The former aims to predict the center word, given the context, whereas the latter aims to predict the context given the center word. The term Word2Vec is often used synonymously for the latter model variant.

10.2.1 *Distributional Semantics*

The overall goal of the skip-gram model is to embed words into a low-dimensional real-valued space, while capturing semantic similarities among words. The model seeks a projection $\phi : w \in V \rightarrow \mathbb{R}^{|V| \times d}$, for every word w in the vocabulary V, where $d << |V|$. The semantic similarity is defined by the distributional hypothesis "you shall know a word by the company it keeps" [53], which states that words in similar contexts tend to have similar meanings [68]. Given a word w, the model aims to estimate the probability for another word u to appear in the context of w, i.e., $P(u|w)$. The context is defined by a sliding window moved over sentences, as illustrated in figure 10.1. In this

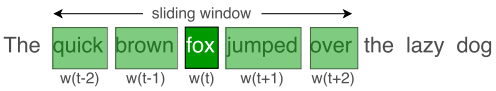


Figure 10.1: Illustration of word embedding sliding window with window size 2.

illustration, the window size is set to 2, i.e., two words to the left and two words to the right are considered as context for the center word. The center word is "fox" and the context words are "quick","brown" (left window) and "jumped","over" (right window).

10.2.2 *Learning Embeddings via Shallow Neural Networks*

As just mentioned, the model's aim is to estimate $P(u|w)$, the probability of word u to appear in the context of a given word w. Let w_i be

the center word of the sliding window and m the window size. The log probability of contextual words is then given by:

$$\sum_{i=1}^{|V|} \sum_{-m<j<m} log(P(w_{i+j}|w_i)) \tag{10.1}$$

In order to retain the probability estimate, the model is trained to maximize the probability of words in context, conditioned on the projection of the center word

$$\sum_{i=1}^{|V|} \sum_{-m<j<m} log(P(w_{i+j}|\phi(w_i))) \tag{10.2}$$

By the law of total probability, the probability of out-of-context words needs to decrease accordingly when maximizing the probability of words in context.

Technically, this is implemented by a shallow neural network with a single linear hidden layer. In the input layer, each node represents a word from the vocabulary and the same applies to the output layer. The mapping ϕ of a word to the embedding space is then given by the weights between a particular input node (word) and the hidden layer. That is, the weights define a word's representation in embedding space. The weights between hidden layer and output layer define the contextual mapping ϕ', often referred to as projection (opposed to embedding). The conditional probability of a single training center-context word pair is then modeled as a softmax unit parametrized by the dot product of the words' feature representations (projections)

$$P(w_{i+j}|\phi(w_i)) = \frac{exp(\langle \phi'(w_{i+j}), \phi(w_i) \rangle)}{\sum_{v \in V} exp(\langle \phi'(v), \phi(w_i) \rangle)} \tag{10.3}$$

where $\langle \cdot \rangle$ is the dot product. Since the normalization in the denominator is costly to compute, Mikolov et al. [133] proposed to replace $logP(w_{i+j}|\phi(w_i))$ by negative sampling:

$$log\ \sigma(\langle \phi'(w_{i+j}), \phi(w_i) \rangle) + \sum_{k=1}^{K} \mathbb{E}_{w_k \sim P_n(w)} [log\ \sigma(-\langle \phi'(w_k), \phi(w_i) \rangle] \tag{10.4}$$

where $\sigma(x) = \frac{1}{1+exp(-x)}$ is the logistic sigmoid function, K is the number of negative samples, drawn from the noise distribution $P_n(w)$, which corresponds to the unigram distribution of words. The negative sampling objective is to distinguish between nodes in the context of w_i and nodes not in the context (negative samples w_k).

Replacing $P(w_{i+j}|\phi(w_i))$ by negative sampling is possible, as the model's aim is not estimating actual probabilities, but finding good representations $\phi(w)$. In fact, negative sampling provides better embeddings than the more precise estimate of the softmax [133].

10.2.3 *Further Optimization Steps*

Mikolov et al. [133] introduced additional heuristics, which they showed empirically to improve the performance (in terms of runtime and/or quality of the embeddings). Those are listed in the following.

REMOVAL OF RARE WORDS A hyper-parameter is used to define the minimum occurrences of words (default value: 5). Words that occur less in a dataset with up to several billions are most likely of no interest or even misspelled words.

NOISE DISTRIBUTION As choice for the noise distribution to draw negative samples from, the unigram distribution raised to the power of $\frac{3}{4}$ showed the best performance (in particular better than unigram and uniform).

SUB-SAMPLING Frequent words are discarded with a probability of $P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$ where $f(w_i)$ is the frequency of word w_i and t a threshold, typically around 10^{-5}. This sub-sampling counters the imbalance between frequent and rare words and reduces the training time by minimizing the number of training pairs.

REDUCED WINDOW SIZE Context words that are closer to the center word should be weighted stronger. Therefore, the actual window size is randomly picked between 1 and the window size m for each center word.

11

OPTIMIZING CONVERGENCE OF SKIP-GRAM WITH NEGATIVE SAMPLING

Since the introduction of skip-gram with negative sampling (SGNS), plenty implementations of it have been made available, not least the original c-implementation by the authors themselves. Most of them aim to ease usability by porting the implementation to a different programming language and/or improve runtime. On the other hand, neural network training today is typically done on GPUs and frameworks like TensorFlow[1] or PyTorch[2] are readily available. Those frameworks provide implementations of common methods for training neural networks, greatly decreasing implementation effort.

The motivation for this chapter is first of all a practical one: We aim for a flexible SGNS implementation which can be easily altered and extended. The above-mentioned frameworks are a perfect fit for such a flexible implementation as for example changing the model's optimizer equates changing a single line of code when using such frameworks. However, the runtime of an implementation in such a framework will definitely be higher than a hand-tuned model-specific implementation. The usual training algorithm for SGNS is stochastic gradient descent, updating the embeddings one after another. A lot of effort has been dedicated to decrease the runtime of SGNS by increasing the throughput without changing the training algorithm. That is, the aim is to increase the amount of words processed per second while still using stochastic gradient descent as training algorithm.

We address the aim to decrease runtime from a different perspective: Instead of increasing the *throughput*, we aim to optimize the *convergence*. If the model converges to high quality embeddings within fewer iterations, the runtime is also reduced accordingly. The throughput optimization could still be applied to a model that converges faster as an additional step, decreasing runtime even further.

11.1 RELATED WORK

One of the most popular skip-gram implementations is the python-port of the original c-code by Radim Řehůřek [153], integrated in Gensim [154]. By re-writing the training loop in Cython, making use of BLAS routines [21] and sigmoid tables, Řehůřek achieved to increase the throughput over a plain NumPy implementation by a factor of ≈73 and over the original c-code by ≈3.5. Řehůřek reports

1 https://www.tensorflow.org – last accessed March 2020
2 https://pytorch.org – last accessed March 2020

a throughput of around 100k words/s, a value we were well able to reproduce up to a reasonably close range on different CPUs.

Both, the implementation by Řehůřek and the original implementation by Mikolov et al. [132] use stochastic gradient descent, which is a sequential algorithm. The sequential nature does not favor parallelization. To deal with this issue, both authors use a Hogwild [152] approach. The idea of Hogwild is to allow multiple threads working independently on different parts of the data. While those threads work independent, they all have access to a shared memory, in this case the single model. Concretely, the dataset is split into N evenly sized chunks and each of these chunks is processed independently by one thread. Since words may occur in multiple of those chunks, collisions are bound to happen if two threads update the embedding of the same word at the same time. According to Recht et al. [152], overwriting errors do not lead to a significant loss in accuracy if the data is sparse enough. However, Vuurens et al. [206] argue that collisions may be be more likely for natural language, given the skew towards frequent words. Frequent words are shared with many words as context, leading to collisions.

The same authors propose a caching scheme in order to reduce updates and therefore reduce collisions. They apply the caching scheme to hierarchical softmax with a huffman tree, instead of skip-gram with negative sampling. The hierarchical softmax is even more prone to collisions as the root node of the tree is in every word's path. Their proposed scheme caches updates of the most frequent nodes in the tree and flushes the cache after a predefined number of steps (10 in their evaluation). That is, instead of immediately updating the embeddings, updates are accumulated and applied after the number of defined steps, therefore reducing concurrent access (and collisions accordingly). Vuurens et al. empirically demonstrated the effectiveness of their caching scheme (see figure 11.1 on the facing page). While the runtime for implementations without caching decreases until a certain amount of threads (8 for Word2Vec (w2v) and c0, 18 for Gensim), it regresses afterwards. Caching the 31 most frequent nodes (c31) further decreases the runtime when increasing the number of threads.

Ji et al. [84, 85] also reduce collisions by exploiting the locality of updates: The same context word is used for model updates of several close-by center words. Their optimization is applied to the skip-gram model. In this model, the locality is lost when calculating the dot-product (cf. equation (10.4) on page 145) and applying updates sequentially for each center-context training pair. The primary goal of Ji et al. was to lift the dot-product, which is a level-1 BLAS [21] operation (vector-vector multiplication) to a level-3 BLAS operation (matrix-matrix multiplication), which is implemented more efficiently. In order to achieve this, they aggregate several center words for the same context word into a matrix. Similarly, they formed the second

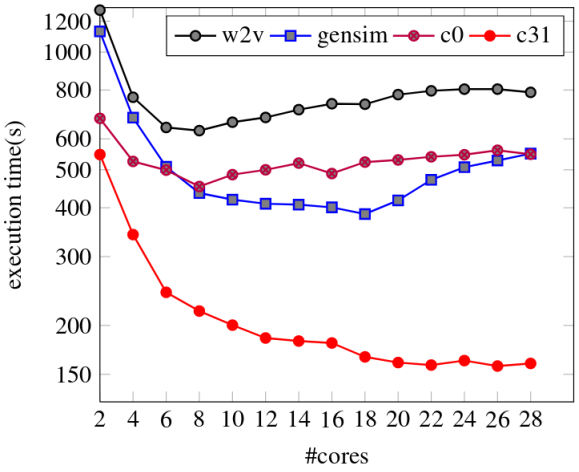


Figure 11.1: Comparison of the execution time in relation to the number of used cores [206, figure 1].

matrix by aggregating the context word and negative samples. The latter aggregation results in the negative samples being shared among several center-context pairs, opposed to the original implementation, where a set of negative samples was drawn individually for each center-context pair. With the transformation to a level-3 BLAS operation the training is organized in mini-batches with a maximum size of twice the context window + 1. By training in mini-batches, the resulting updates are aggregates of the batch, hence decreasing potential collisions. The authors report a 2.6x speedup over the original c-implementation in a single thread and almost perfect linear scaling by increasing threads on a single CPU socket. Due to inter-socket communication overhead, the runtime scales sub-linear with increasing threads on two sockets. The authors' evaluation showed hardly to no negative impact on the quality of the resulting embeddings.

Besides optimizing the throughput on CPUs, several works addressed throughput optimization on GPUs. The key advantage of GPUs are their massive parallelization capabilities. However, in order to exploit those parallelization capabilities, the skip-gram algorithm needs to be adapted to GPU architectures accordingly. Seulki and Yi [9] achieve this adaptation by processing sentences in parallel among CUDA[3] blocks and parallelizing by threads within a block over the embedding dimensions. The authors report a throughput of approximately 180k words/s on a single GPU without a loss in the quality of the produced embeddings. This strategy is also pursued by Gupta and Khare [65]. The latter report a throughput of around 13 million words/s when parallelizing over 8 GPUs and only a small drop in embedding quality when parallelizing over multiple GPUs. Canny et al. [31] identify memory access as bottleneck and similar to

3 https://developer.nvidia.com/cuda-zone – last accessed March 2020

the above-mentioned approaches merge multiple memory accesses into one. They report a throughput of almost 9 million words/s on a single GPU. They did not present evaluations about the quality of the resulting embeddings. When trying to reproduce the work, other researches reported poor quality of the embeddings[4].

11.2 APPROACH

A common phenomenon we observe in related work is the aim to increase parallelism and decrease collisions due to concurrent updates of the same embedding. To achieve this aim, most authors exploit specific properties of the skip-gram model and the hardware architecture (CPU or GPU). This is contradictory to our aim of a flexible implementation that can be easily altered or extended. A model/architecture specific implementation may no longer be valid if the model is extended and requires a new specific implementation. In the worst case, the extended model needs to be implemented again from scratch, resulting in high implementation time.

We also observe, that all of the approaches described in the previous section 11.1 on page 147 are build around stochastic gradient descent as training algorithm. Recently, advanced optimizers such as AdaGrad [48] or Adam [95] have been proposed. Both these algorithms improve the updates of gradient descent by taking past gradients into account. AdaGrad accounts for the complete history, whereas Adam uses a moving average. The updates at the current timestep are scaled by a factor corresponding to the magnitude of past gradients (in addition to scaling by learning rate). This is particularly relevant for the skip-gram model, as frequent words receive many updates. Therefore, scaling by the magnitude of past gradients scales down the updates of frequent words, balancing between frequent and rare words. We opt for Adam as optimizer, as with AdaGrad the update steps are constantly scaled down as the training progresses, which may result in the training process to stall.

For the implementation, we use PyTorch [144], which describes itself as "a replacement for NumPy to use the power of GPUs"[5]. While PyTorch models can be executed on the CPU as well, this choice implies the use of GPUs for performance reasons, as a NumPy implementation of the skip-gram model is around 70x slower than the Gensim implementation for example (cf. section 11.1 on page 147). We opted for PyTorch as an easy-to-use, highly flexible framework as its models are just Python programs, computations are executed dynamically and its share is on the rise (mentioned in over 40% of papers that mention common deep learning frameworks on arXiv) [144].

4 https://github.com/BIDData/BIDMach/issues/96 – last accessed March 2020
5 https://pytorch.org/tutorials/beginner/blitz/tensor_tutorial.html#what-is-pytorch – last accessed March 2020

With the use of GPUs comes the need for parallelism. The typical method of training neural networks is to train in batches, i.e., partitioning the dataset into (small) chunks. We follow this approach, as within a batch, computation can run in parallel. Concretely, a batch in our model consists of b training pairs, i.e., b center words, b corresponding context words and $b *$ negative samples. Equation (10.4) on page 145 is calculated in parallel for all b pairs in the batch. If a center (or context word) occurs multiple times within a batch, its updates are accumulated. Therefore, collisions are avoided and oscillations (if a context word draws the embedding in one direction, while another context word draws it in the opposite direction) are smoothed out. On the downside, all updates in the batch are based on the initial state of the weights. In contrast, in a sequential gradient descent implementation, a word could have received up to $b - 1$ updates before the current update step, compared to the initial batch state.

11.2.1 *Batch Diversity*

To counter the initial state issue, words in a batch should be as diverse as possible. In the ideal case, a word would be contained in a batch only once (either as center word, context word or negative sample). Then it would not suffer from missing previous updates. In the following, we discuss different strategies to increase diversity within a batch per word role (center, context or negative sample).

11.2.1.1 *Center Words*

For center words, we distinguish between a fixed batch size and a variable batch size.

FIXED BATCH SIZE When the batch size is fixed, the simplest option to increase diversity is to generate batches only from distinct center words at the beginning greedily. However, full diversity in the first batches leads to degenerate diversity in the latter batches as they are filled with the most frequent words. Balancing center word diversity across all batches is an optimization problem of its own, which comes at a high computational cost and therefore is an unsuitable approach.

VARIABLE BATCH SIZE A variable batch size allows to fill each batch with only distinct center words, with the batch size equal to the number of distinct words. This results in a few very large batches in the beginning and a massive amount of tiny batches as the training continues, due to a small set of most frequent words making the majority of words in the training corpus. As parallelism happens within a batch, small batch sizes are detrimental to the runtime, rendering this approach also unsuitable. We therefore ignore a variable batch size in the subsequent discussion.

11.2.1.2 *Context Words*

Context words exhibit the same behavior as center words. With a greedy approach, full diversity is possible in the first batches, but comes at the cost of degenerate diversity in latter batches. An optimal distribution of context words across all batches comes at too high computation costs and still cannot guarantee full diversity in each batch.

11.2.1.3 *Negative Samples*

Diversity among negative samples could be achieved by drawing samples without replacement. However, by drawing without replacement, the noise distribution is altered. Further, negative samples have the additional constraint that they should differ from context words, since both are represented as embeddings in the projection layer of the network. A word in the projection layer can either act as a negative sample or a context word. With this additional constraint, the discussion of center and context words applies equally to negative samples.

11.2.1.4 *Shuffling*

Given the above-mentioned issues, we conclude that the best achievable diversity is too costly to calculate and high diversity in early batches comes at the cost of degenerate diversity in later batches. We therefore shuffle all training pairs before assembling the batches, in order to increase diversity within a batch. While this approach cannot guarantee optimal diversity for any of the batches, it can be seen as a trade-off between high diversity in early batches and degenerate diversity in later batches.

11.2.2 *Regularization*

Mu et al. [135] showed that while various word vectors can result in the same objective value (i.e., the same model loss), not all of them are equally valuable. They further show quadratic (ℓ_2) regularization to provide a viable solution to restrict the word embeddings to high-quality representations. Several other authors made similar observations when applying the skip-gram model to documents [4] or networks [227, 228]. All of them report an increase in performance on some downstream learning task after a few training iterations, but decreasing performance as the training progresses afterwards (without regularization). The evaluations of Zhang et al. [227, 228] reveal the norms of frequent center and context word embeddings to increase in the beginning and then tending to converge. While the embedding norms of rare words are smaller than the frequent ones in the beginning, they continue to grow, ultimately surpassing the norms of frequent embeddings. This crossing point roughly coincides

with the drop in performance. The authors show that ℓ_2-regularization prohibits continuous growth and keeps the relative ordering (rare embeddings have smaller norms than frequent embeddings). When applied to both, embedding and projection layer (i.e., center and context words/negative samples), they show ℓ_2-regularization to stabilize the training, retaining a constantly high performance.

We follow their approach and add ℓ_2-regularization to both, embeddings and projections. As in related work, we use the squared ℓ_2-norm. Let θ be the trainable parameters of the projection in the embedding layer $\phi(x)$ and θ' the trainable parameters of $\phi'(x)$. Then we can represent the projection of words in the in- and output layer by their embedding and projection vectors θ_i and θ'_i. We rewrite equation (10.4) on page 145 as

$$log\ \sigma(\langle \theta'_{i+j}, \theta_i \rangle) + \sum_{k=1}^{K} \mathbb{E}_{w_k \sim P_n(w)} [log\ \sigma(-\langle \theta'_k, \theta_i \rangle] \tag{11.1}$$

Adding regularization we arrive at

$$\begin{aligned} &log\ \sigma(\langle \theta'_{i+j}, \theta_i \rangle) + \sum_{k=1}^{K} \mathbb{E}_{w_k \sim P_n(w)} [log\ \sigma(-\langle \theta'_k, \theta_i \rangle] \\ &\quad + \lambda \left\| \theta_i \right\|_2^2 + \lambda \left\| \theta'_{i+j} \right\|_2^2 + \lambda \sum_{k=1}^{K} \left\| \theta_k \right\|_2^2 \end{aligned} \tag{11.2}$$

where $\|\cdot\|_2$ is the ℓ_2-norm and λ a weighting factor. As all regularization factors share λ as weighting factor, we can rewrite the regularization term as

$$\lambda \left(\left\| \theta_i \right\|_2^2 + \left\| \theta'_{i+j} \right\|_2^2 + \sum_{k=1}^{K} \left\| \theta_k \right\|_2^2 \right) \tag{11.3}$$

which is equivalent to an ℓ_2-penalty on the individual weights, as the partial derivatives with respect to a particular weight are the same for ℓ_2-penalties on individual weights and the squared ℓ_2-norm as defined in equation (11.3). For example $\left\| \theta_i \right\|_2^2$ is equivalent to $\sum_{z=1}^{d} \left(\theta_i^z \right)^2$ where d is the dimensionality of the embedding vector and θ_i^z indicates the z-th dimension of vector θ_i. The regularization term in a batch is then simply the sum over all involved squared weights.

11.3 EVALUATION

For the evaluation, we first train embeddings on the text8[6] dataset which consists of roughly the first 17 million words of Wikipedia (dump from 2006). We then evaluate the quality of the learned embeddings on a word similarity task, with the WS-353 dataset [52]. The

6 http://mattmahoney.net/dc/text8.zip – last accessed March 2020

WS-353 contains human judgments on the similarity of word pairs. We rank the pairs by cosine similarity of the corresponding embeddings and measure the agreement with the human annotation by Spearman's rank correlation coefficient. As this setup is a popular pipeline for word embedding evaluation, it allows to compare evaluation results with the results obtained by different authors. Still, attention has to be paid to the individual setup, as there are slight variations. Some authors report Pearson's correlation coefficient instead of Spearman's rank correlation coefficient. The Pearson correlation coefficient is typically a bit smaller than the latter. Also, text8 might be replaced by a different (potentially larger) dataset, which has an influence on the embedding quality. Last but not least, WS-353 can be divided into two subsets, that distinguish between similarity and relatedness [3]. Some authors report evaluation results only for the similarity subset. Our results are obtained on the full WS-353 dataset. Reported scores in related work range from 0.64 to 0.71 (taking both, Pearson and Spearman, into account).

HARDWARE The experiments are performed on a dual socket server equipped with Intel Xeon Gold 5115 CPUs @ 2.40GHz with 10 cores (20 threads) and two Nvidia Tesla P100 GPUs. While two GPUs are present in the server, all experiments are performed on a single GPU.

11.3.1 *Experimental Setup*

We compare our model (in different variants) against Gensim on the evaluation task described above. The variants comprise Adam and stochastic gradient descent as optimizers and whether training pairs are shuffled before batch creation or not. Results are averaged over 10 runs of each model and we used the following hyperparameters:

DIMENSIONS $d = 100$ The number of dimensions in the words' vector representations. We use 100 dimensions for all models, both in the embedding and projection layer.

NEGATIVE SAMPLES $k = 10$ The amount of samples to draw from the noise distribution. We draw 10 negative samples per center-context pair for all models. We enforce negative samples not to be the same as the context word for comparability reasons, as Gensim applies this restriction as well.

WINDOW SIZE $m = 5$ The amount of context words within the sliding window, left and right to the center words. With $m = 5$, 5 words to the left and 5 words to the right are considered as context. The value is the same for all models.

INITIAL LEARNING RATE α We set the initial learning rate to $\alpha = 0.025$ for stochastic gradient descent (Gensim default) and $\alpha =$

0.001 for Adam (PyTorch default). We linearly decay the learning rate per epoch until α_{min} (see below) in the last epoch.

MINIMUM LEARNING RATE $\alpha_{min} = 0.0001$ Learning rate of the last epoch. The initial learning rate is linearly decayed to this value. The minimum learning rate is the same for all models.

NUMBER OF EPOCHS We train all models for 10 epochs, i.e., 10 iterations over the whole dataset.

SAMPLING THRESHOLD $t = 0.0001$ Threshold for sub-sampling frequent words (cf. section 10.2.3 on page 146).

BATCH SIZE b We train all our model variants with a batch size of $b = 2000$, except for stochastic gradient descent (SGD) without shuffling where we reduce the batch size to $b = 500$.

ℓ_2-REGULARIZATION WEIGHT $\lambda = 1e-6$ The weighting factor of the ℓ_2-regularization term. ℓ_2-regularization is applied to all model parameters, both in the embedding and projection layer in our model variants.

11.3.2 *Results and Discussion*

Figure 11.2 shows the evolution of the word similarity scores of all model variants and Gensim as the training progresses. Gensim does

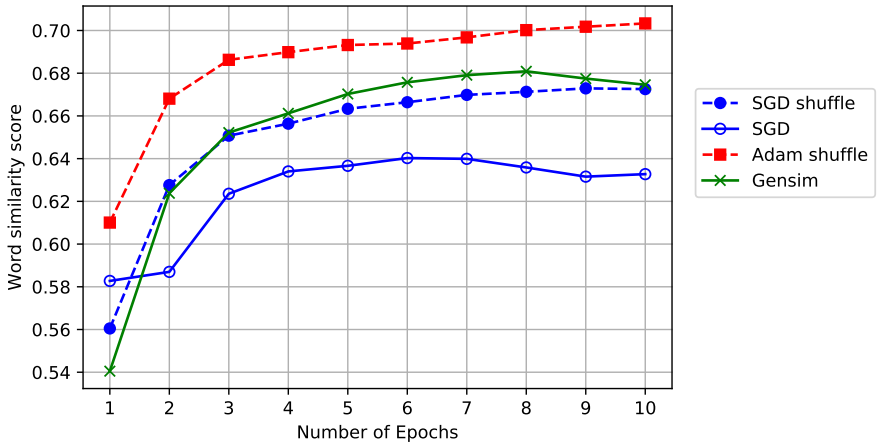


Figure 11.2: Spearman's rank correlation coefficient between model scores and human judgment on WS-353 dataset for word similarity. The graphs show the models' evolution over 10 epochs when trained on the text8 dataset.

not apply ℓ_2-regularization and we can observe the issue of increasing performance in the beginning, but decreasing after a certain point as the training progresses as mentioned in related work (cf. section 11.2.2 on page 152). We also observed this behavior when training for 20

epochs. In contrast, the model scores with ℓ_2-regularization are stable, without this drop in performance, except for stochastic gradient descent (SGD) without shuffling. The training for the latter is rather unstable in general. In fact, we had to reduce the batch size to 500 for this approach to learn meaningful representations at all. With a batch size of 2000, this model oscillated around similarity scores of 0.3.

The models with shuffling consistently outperform the models without and in particular, when combined with the Adam optimizer, the scores are well above the Gensim implementation. This justifies the choice of random shuffling over explicitly maximizing the batch diversity (cf. section 11.2.1 on page 151). Apparently, random shuffling provides sufficient diversity to arrive at high quality embedding vectors.

Within 3 epochs, our model with the Adam optimizer, shuffled batches and ℓ_2-regularization already outperforms the best score achieved by Gensim after 8 epochs. It therefore converges faster and to a better score than the stochastic gradient descent baseline.

In terms of throughput, our model can process around 40k words/s, opposed to Gensim's 700k words/s on the same machine (18 threads). Therefore, even though our model converges faster, it still requires almost seven times as much time as Gensim to arrive at a comparable score. A quick profiling of the runtime revealed that preparing the batches accounts for the majority of computation time (62%), i.e., the GPU has to wait for training data to arrive. Optimizing the batch preparation could reduce the runtime to almost up to the time needed for batch preparation. CUDA calls are asynchronous, i.e., during the actual model computations on the GPU, the next batch could be prepared on the CPU. Only then, the training could be parallelized over multiple GPUs (parallelization is not meaningful if a single GPU already has to wait for training data). Increasing the batch size could further decrease runtime, as the transfer of few large chunks of data to the GPU is beneficial over many smaller chunks. Increasing batch size in our implementation resulted in an additional increase of batch preparation time, therefore harming overall runtime (but without an apparent loss in word similarity scores). An optimized batch preparation could remedy this issue.

However, optimizing the batch preparation is detrimental to our goal of a flexible implementation, as it would tie the batch preparation to this particular model. The benefit of a flexible implementation becomes obvious in the choice of optimizers: Replacing stochastic gradient descent by Adam only requires to change a single line of code and further optimizers are readily available in PyTorch. As PyTorch is under active development future methods are highly likely to be integrated in the framework, eliminating the need for an own implementation. The ℓ_2-regularization provides a further argument for the flexible implementation: By our choice of the regularization term,

ℓ_2-regularization on the word vectors is equal to an ℓ_2-penalty on the individual weights (cf. section 11.2.2 on page 152). The latter is readily available in PyTorch and only requires to alter a single parameter value.

Populating the batches with randomly shuffled training pairs comes at the cost of increased memory consumption. With shuffling, all the pairs need to be kept in main memory, whereas with a sequential approach it suffices to move the sliding window over the input data until the current batch is filled with training pairs. If the available memory is limited, the data could be partitioned into chunks, such that all training pairs of a single chunk fit into memory. The model could then be trained on the chunks sequentially training each chunk with the same procedure as the whole dataset.

11.4 CONCLUSION

The evaluation results show that our model with ℓ_2-regularization, shuffling and Adam optimizer converges faster and to better scores than the original baseline, represented by the Gensim implementation. While converging to state-of-the-art scores in few epochs, our goal of a flexible implementation is detrimental to the throughput. The popular Gensim implementation has an almost 18 times higher throughput than our model. We therefore conclude that our implementation is suitable for prototyping, allowing to easily alter and extend the model and therefore quickly investigate multiple model variants or extensions. Once a promising model is found, this model could still be optimized for throughput.

12 NETWORK EMBEDDING

Graph analysis involves predictions over nodes, edges and further network properties, such as for example shortest paths. A prominent example of predictions over nodes is node classification, i.e., predicting the label(s) of a node. In a social network, we might for example predict the interests of a user or the community, this user belongs to. Analogously, link prediction aims to identify, whether a connecting edge should exist between a pair of nodes. In a social network for instance, link prediction can be used to discover novel connections, which are most likely to be established, i.e., users making friends. In this work, we focus on the the node classification task, as it most relevant to our goal of a unified representation of resources. Node classification can be seen as a proxy task for accessing resources via keywords.

Network embedding aims to find meaningful feature representations (embeddings) of nodes, edges or even whole (sub-)graphs to be used as input in the aforementioned downstream machine learning tasks. A common graph-specific objective is to preserve the local neighborhood of a node, when learning feature representations for nodes. With DeepWalk, Perozzi et al. [147] proposed an embedding approach to preserve this local neighborhood that leverages the skip-gram model with negative sampling (cf. section 10.2 on page 144). They sample truncated random walks from the graph and treat them as sentence equivalents in the skip-gram model to learn node representations. However, when optimizing for the local neighborhood, the global structure might be lost in the feature representations. For the random-walk based model as introduced by Perozzi et al., several extensions to local optimization have been proposed, which aim to retain global structure. These extensions comprise adaptations of the random walk behavior itself and abstraction layers of the graph (either encoded in the random walks or sampling random walks at different graph abstraction layers).

In this chapter, we investigate various extensions to the random-walk based model and complement them with our own additional methods. After refining the problem definition of network embedding and providing background information on DeepWalk in the next section 12.1 on the next page, we review related work in the subsequent section 12.2 on page 161. When then first investigate and evaluate means to control the random walk behavior in section 12.3 on page 162 and afterwards abstraction layers of the graph in section 12.4 on page 168. Finally, we conclude the chapter in section 12.5 on page 174.

12.1 BACKGROUND

We start this section with a definition of the network embedding goal. Please note how the problem definition is similar to the word embedding objective defined in section 10.2.1 on page 144. We then provide background information on the DeepWalk method and its relation to the skip-gram model with negative sampling.

12.1.1 *Problem Definition*

Let $G = (V, E)$ denote the graph, where V is the set of nodes and E the set of edges, $E \subseteq (V \times V)$. The goal is to find an embedding for the nodes (or equivalently for the edges) $\phi : v \in V \rightarrow \mathbb{R}^{|V| \times d}$, where $d << |V|$. We want to embed the nodes into a lower-dimensional, real-valued space, while still retaining as much information as possible as in the original space.

12.1.2 *DeepWalk*

To learn the mapping defined above, Perozzi et al. [147] seek to estimate the likelihood of a node u to appear in the context (i.e., the neighborhood) of another node v. Similar to the word embedding approach by Mikolov et al. [133], given an embedding of a node ($\phi(v)$) they seek to maximize the log likelihood of nodes in the neighborhood given by

$$\sum_{i=1}^{V} \sum_{-m<j<m} log(P(v_{i+j}|v_i)) \tag{12.1}$$

This definition is equal to the word embedding objective (cf. equation (10.1) on page 145) with replacing words w by nodes v. Here V refers to the set of nodes in the graph, whereas V referred to the vocabulary of all words in equation (10.1) on page 145.

The subsequent steps and reasoning of the skip-gram model with negative sampling for word embeddings equally apply to node embeddings. Therefore we arrive at the following negative sampling objective for a single center-context node pair

$$log\ \sigma(\langle \phi'(v_{i+j}), \phi(v_i) \rangle) + \sum_{k=1}^{K} \mathbb{E}_{v_k \sim P_n(v)} [log\ \sigma(-\langle \phi'(v_k), \phi(v_i) \rangle)] \tag{12.2}$$

Again, this is the same as equation (10.4) on page 145, only words w are replaced by nodes v and the noise distribution corresponds to node frequencies, which we will detail shortly.

Until now, we only defined context as the neighborhood of a node. More precisely, contextual nodes are derived from sampling truncated random walks from the graph and treating them as sentence equivalents. Accordingly, a sliding window moves over the sampled walks

and nodes within this window are considered as context for the center node. The underlying idea is similar to the distributional hypothesis in that nodes that share a similar neighborhood are considered similar. For sampling the walks, a pre-determined set of walks with fixed length is started at each node of the graph, which then randomly traverses the graph along the nodes' adjacent edges. The probability to traverse to the next node along a particular edge is given by the inverse node degree of the current node v ($\frac{1}{deg(v)}$). The noise distribution for negative sampling corresponds to the nodes' frequency in the sampled walks. Additional optimization steps of the skip-gram model with negative sampling for word embedding (sub-sampling, removal of rare words, ... cf. section 10.2.3 on page 146) equally apply to node embedding.

12.2 RELATED WORK

Graph embeddings can be obtained by either applying a general dimensionality reduction algorithm or by methods that are specifically tailored towards network-specific properties. A wide variety has been proposed in the literature (cf. Goyal and Ferrara [61] for a survey). Among the classical methods are Principal Component Analysis (PCA) [88], Linear Discriminant Analysis (LDA) [127], ISOMAP [202], Multidimensional Scaling (MDS) [104], LLE [159] and Laplacian Eigenmaps [15] (cf. Yan et al. [219] for a survey). Most of these methods typically rely on solving eigen decomposition and the complexity is at least quadratic in the number of nodes, which makes them inefficient to handle large-scale networks.

DeepWalk is one of the first methods to leverage the skip-gram model with negative sampling [108] to graphs via truncated random walks. Similar to Word2Vec implicitly factorizing a matrix of word co-occurrences [115, 145], DeepWalk has been shown to factorize a matrix of node transition probabilities [220].

Node2Vec [62] extends DeepWalk by introducing parameters to control the random walk behavior, aiming to discover not only neighborhood similarities (homophily) but also structural roles of nodes (structural equivalence).

LINE [198] explicitly optimizes the embeddings to capture first- and second-order proximity, by training separate embeddings for them, which are finally concatenated. First-order proximity is given by explicit connections between nodes, while second-order proximity is given by comparing the nodes' neighborhoods. Liu et. al build on LINE and present a method that is capable to embed large-scale graphs distributively in a streaming fashion [123].

Instead of approximating the k-order proximity matrix, as DeepWalk does, GraRep [32] calculates it accurately, at the cost of increased complexity. Yang et al. [221] alleviate this problem by using informa-

tion from lower order proximity matrices. The authors of HOPE [140] experimented with different similarity measures, such as Katz Index, Rooted Page Rank and Adamic-Adar.

HARP [37] and Walklets [146] address capturing higher-order proximity by adapting the random walk strategy. While HARP coarsens the graph and learns representations via hierarchically collapsed graphs, Walklets skips over steps in the random walks. Compared to DeepWalk or Node2Vec, both HARP and Walklets exhibit additional complexity, as the node representations are learned on multiple levels - The collapsing level in HARP and the skip level in Walklets. More details on those two methods will be provided in section 12.4.2 on page 169.

Besides most of these approaches utilizing shallow neural network architectures to learn the feature representations, deep architectures have been proposed as well, aiming at capturing non-linearity in the graphs. SDNE [207] and DNGR [33] utilize autoencoders, GCN [97] defines a convolution operator on the graph. Further, methods that incorporate additional information, such as particular graph properties, e.g. communities [92, 210] or node attributes [78, 220] have been proposed. While random walk based methods in principle can incorporate the direction of edges during the random walk, this asymmetry is not encoded in the final embeddings (due to the symmetry of the dot product). Khosla et al. [94] proposed an approach to maintain the different roles of nodes, according to the direction of edges.

The focus of this chapter is on approaches centered around random walks, i.e., DeepWalk, Node2Vec, HARP and Walklets. Experimental results reported in different papers are often hard to compare, due to varying experimental setups, evaluation metrics or datasets. Nevertheless, the results reported by random walk based methods deliver state of the art performance in tasks such as node classification and can compete with even far more complex models. We will provide more details on the particular aforementioned random walk based methods in the corresponding sections, when we investigate means to control the random walk behavior (next section 12.3) and abstraction levels (section 12.4 on page 168).

12.3 CONTROLLING RANDOM WALK BEHAVIOR

Node2Vec as proposed by Grover and Leskovec [62] extends DeepWalk by introducing two additional parameters to guide the random walk, aiming at preserving both community structure and structural roles. Community structure in a graph is based on proximity, i.e., nodes that are close together belong to a community. Structural roles can be nodes that act as bridges between sub-networks, or hubs, which are the main exchange point between many nodes. The two parameters guiding the random walk in Node2Vec are:

- Return parameter p, controlling the likeliness of immediately revisiting a node.
- In-out parameter q, controlling how far outward the random walk should progress from the starting node.

These parameters allow to resemble depth-first (DFS) or breadth-first (BFS) search-like behavior in the most extreme setting, as well as a smooth interpolation between DFS and BFS. Grover and Leskovec suggest "that BFS and DFS strategies represent extreme ends on the spectrum of embedding nodes based on the principles of homophily (i.e., network communities) and structural equivalence (i.e., structural roles of nodes)" [62]. In a case study, they demonstrate that subject to the parameter settings, the resulting embeddings can capture homophily or structural equivalence.

In this section, we investigate the influence of different random walk strategies on the resulting embeddings. In particular, we

- try to reproduce the case study illustrating homophily and structural equivalence.
- try to reproduce Node2Vec's node classification result.
- introduce two additional modifications to the random walk strategy and evaluate and compare them on the node classification task. The additional strategies comprise hub attention and jump probabilities, where the latter can be seen as noise.

12.3.1 *Random Walk Modifications*

The modifications we introduce to the random walk strategy (hub attention and jump probabilities) are similar to the one implemented by Node2Vec [62]. Modifications can take place in two sections of the random walk algorithm: During *sampling*, we can modify the transition probabilities between nodes to draw attention to specific ones, while during *walking*, we can directly influence how the random walk traverses.

12.3.1.1 *Jump Probability*

We introduce the parameter j to modify the random walk during walking. It controls the probability of jumping to a random node in the graph at any given time. Intuitively, j ranges from 0 to 1, where with $j = 0$ no jumps to a random node occur and with $j = 1$ every walking step is a random jump. The latter allows to traverse the graph truly random, without drawing any attention to the structure of the graph, ignoring its edges with their respective weights. We are sampling truncated random walks, i.e., we start walks with a fixed length from every node in the graph. Therefore, the jump probability can be seen

as noise in the truncated walks, opposed to jump probability in a single (huge) walk (as used by PageRank [141] for example).

12.3.1.2 *Hub Attention*

Hubs are nodes in a graph with a degree that greatly exceeds the average [10]. The threshold ε is the sum of the mean node degree and the standard deviation of all degrees:

$$\varepsilon = \overline{deg} + \sigma_{deg}$$

Based on that, we define the subset of hubs H of the nodes V as every node with a higher degree than ε:

$$H = \{v \in V \mid deg(v) > \varepsilon\} \subset V$$

We are now modifying the random walk during sampling. Similar to Node2Vec, we change the unnormalized transition probability π_{vx} of a node v to its neighbors x on edges (v, x) to $\pi_{vx} = \alpha_h(x) * w_{vx}$, where w_{vx} is the weight of the edge, and

$$\alpha_h(x) = \begin{cases} \frac{1}{h} & if\ x \in H \\ 1 & else \end{cases}$$

The parameter h introduced here controls the random walk tendency towards and away from hubs. If this parameter is set to a high value (> 1), the probability of visiting hubs is reduced, and less frequented nodes are explored. On the other hand, if h is small (< 1), it increases the probability of traversing to hubs, so the walk is more focused on the areas around hubs in the graph.

12.3.2 *Evaluation*

We start this section by introducing the datasets and parameters used in our experiments, followed by the reproduction of the case study and the evaluation on the node classification task.

Les Misérables [113] is a network which contains the characters and their co-appearances in the novel "Les Misérables" by Victor Hugo. Every node represents a character, and an edge between two characters indicates that they appeared in the same book chapter. The graph consists of 77 nodes, connected via 254 edges.

BlogCatalog [224] is a social network where every node is a blogger and an edge between two of them represents friendship. The graph consists of 10,312 nodes, connected via 333,983 edges and assigned to one or more of 39 classes (multi-label). The classes are the topics the blogger is interested in.

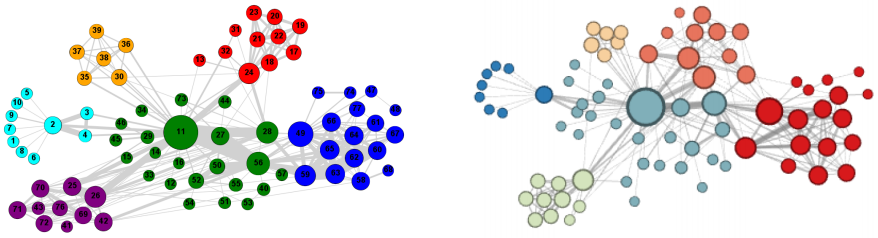
We define a set of parameters for all learning algorithms to create a basis for a fair comparison. These are the embedding dimension d, the

walk length l, the number of walks n and the skip-gram model [108] window size m. Furthermore the algorithm-specific parameters p, q (Node2Vec [62]), h (Hub Attention, cf. section 12.3.1.2 on the facing page) and j (Jump Probability, cf. section section 12.3.1.1 on page 163).

12.3.2.1 *Reproduction of Les Misérables Case Study*

For the reproduction of the case study, we first train embeddings with Node2Vec on the Les Misérables dataset with the respective parameter settings of p and q. We then cluster these embeddings using k-means and assign colors to the nodes in the graph corresponding to their cluster in embedding space. We then visualize the graph with Gephi [11]. For the common parameters, we used the values reported by Grover and Leskovec: embedding dimension $d = 16$, walk length $l = 80$, number of walks $n = 10$ and context window size $m = 10$. The embeddings are trained for five iterations with the learning rate linearly decaying from 0.025 to 0.0001 per epoch. The sub-sampling threshold is set to $t = 0.001$, number of negative samples $k = 5$ and rare nodes are kept. We set the walk parametrization parameters p and q to the values reported in the original paper [62], aimed at either capturing homophily or structural equivalence. The resulting graphs are visualized in figure 12.1 (homophily) and figure 12.2 on the next page (structural equivalence).

For reflecting homophily, Grover and Leskovec report the parameters $p = 1$ and $q = 0.5$. $p = 1$ induces no bias on re-visiting nodes, while with $q = 0.5$ the walks are biased towards global exploration, i.e., tending to walk further away from the current node. With these parameters, we were well able to reproduce their result. Our visualization of the graph in figure 12.1a also reflects community structure, comparable to the visualization of the original paper in figure 12.1b.

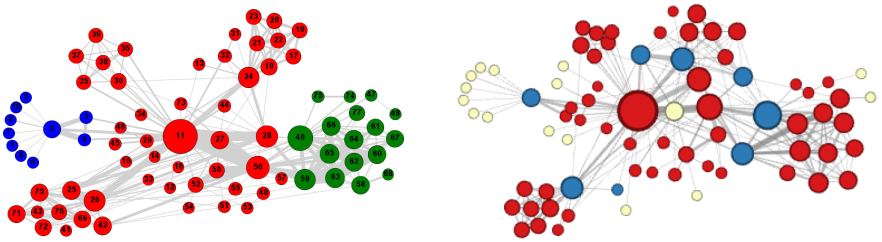


(a) reproduction result (b) original [62, figure 3, upper graph]

Figure 12.1: Les Misérables network homophily showcase. Nodes colored corresponding to their cluster in embedding space, walk parameters $p = 1, q = 0.5$.

In order to reflect structural equivalence, Grover and Leskovec [62] report $p = 1$ and $q = 2$ as parameter setting for the walk behavior. Again, $p = 1$ induces no bias on re-visiting nodes, whereas this time with $q = 2$ the walks are biased towards exploring the local

neighborhood. With these parameters, we were not able to reproduce the reported result. Our reproduction graph in figure 12.2a represents community structure as before (with 3 instead of 6 clusters), but not structural equivalence as the original visualization in figure 12.2b. Even with grid-search over the walk and further parameters (walk



(a) reproduction result (b) original [62, figure 3, lower graph]

Figure 12.2: Les Misérables network structural equivalence showcase. Nodes colored corresponding to their cluster in embedding space, walk parameters $p = 1, q = 2$.

length l, number of walks n and context window size m), no result close to the original could be produced. The visualized graphs always represented community structure instead of structural equivalence.

12.3.2.2 *Node Classification*

For node classification, we run a multi-label node classification task on the BlogCatalog dataset. Same as in the original paper [62], we use a one-vs-rest logistic regression classifier with L2 regularization. We also use an oracle for the number of labels to predict. The dataset is split into training and test data, with a training fraction of 50%, and the scores are averaged over 10 random splits. Again, we use the common parameters as reported by Grover and Leskovec [62]: embedding dimension $d = 128$, walk length $l = 80$, number of walks $n = 10$ and window size $m = 10$. The remaining parameters are the same as for the case study. All results of different configurations are presented in table 12.1 on the facing page. The underlined values are the best per walk strategy, and the bold one is the best value overall.

NODE2VEC To reproduce the results of Node2Vec [62], we used the values $p = q = 0.25$ as reported in the paper. We also included results of Node2Vec with parameters $p = q = 1$, eliminating the influence of the parameters on the random walk, resembling a "pure" random walk like DeepWalk [147]. We were not able to reproduce the results within a single iteration of the skip-gram model (as reported by the authors), but used five iterations instead. We suspect the difference to be due to an unrecognized change of the default hyper-parameters in Gensim[1], the skip-gram with negative sampling implementation used

1 https://radimrehurek.com/gensim/ – last accessed March 2020

Table 12.1: Macro F1 scores and standard deviation (±) of the node classification task on the BlogCatalog dataset with different parameters for each algorithm.

Learner	Parameters	Score
Node2Vec	$p = 0.25, q = 0.25$	$\underline{26.72} \pm 0.72$
	$p = 1, q = 1$	25.85 ± 0.59
Jump Probability	$j = 1$	03.64 ± 0.14
	$j = 0.25$	25.27 ± 0.68
	$j = 0.1$	$\underline{25.76} \pm 0.55$
Hub Attention	$h = 0.5$	23.37 ± 0.59
	$h = 0.75$	25.21 ± 0.64
	$h = 4$	$\underline{\mathbf{27.44}} \pm 0.64$
	$h = 8$	27.17 ± 0.52
	$h = 10$	27.39 ± 0.55

by Node2Vec (cf. section 11.1 on page 147). Our reproduced score is close to the reported one and even slightly better, which can be explained by the random factor of the experiment. From this, we can conclude that our values are consistent with those in the original paper and that we have created a basis for further comparison. Furthermore, Node2Vec [62] performed better than non-parameterized random walks like DeepWalk, at least on this dataset and these parameter settings.

JUMP PROBABILITY For the Jump Probability algorithm, we set j to each of $\{1, 0.25, 0.1\}$. Expectably, the higher the jump probability is, the consistently worse the results get. However, at $j = 0.1$, i.e., 10% noise, the performance is close to DeepWalk. This indicates, that a small amount of noise in the walks does not drastically harm the performance of the resulting embeddings on the node classification task.

HUB ATTENTION For the Hub Attention algorithm we set h to each of $\{0.5, 0.75, 4, 8, 10\}$. As shown in table 12.1, a value of $h = 4$ allows us to achieve the best value across the different strategies. When focusing on hubs ($h = 0.5, h = 0.75$), performance even drops below the score of a jump probability of 25% ($j = 0.25$), i.e., jumping to a random node in every 4th step on average. Conversely, we gain performance if we put more attention to otherwise less frequently visited nodes (i.e., if we increase h), at least up to a certain point where the results stabilize.

12.3.2.3 *Discussion*

We attribute our inability to reproduce the case study in terms of structural equivalence to the skip-gram model. Its objective is to predict neighboring nodes and hence, nodes with similar neighbors are represented closeby in embedding space. Another factor is the context window size of the skip-gram model: No matter how far out a walk traverses, only nodes within this window will be considered as context. This also means, that the walks which start at a particular node are not that relevant to this particular node, but its embedding is determined by all the walks traversing through this node. With optimal parameter settings and taking the standard deviation into account, the performance difference between the walk strategies on the node classification task is negligible. In addition, Perozzi et al. report a macro-F1 score of 27.3 for DeepWalk [147], doing shorter, but more walks. They report performance to increase constantly with the number of walks until it finally stabilizes. Guiding the random walks away from Hubs and therefore putting their focus on otherwise less frequently visited nodes increases the balance between rare and frequent nodes in the walks. The results indicate benefit for the resulting embeddings.

We conclude, that adapting the walk strategy can improve the embedding performance, if the number of sampled walks is insufficient. However, instead of tuning hyper-parameters of particular walk strategies, we can also increase the number of sampled walks per node instead. The nature of the skip-gram model and our inability to reproduce the structural equivalence case study point to the embeddings always representing homophily.

12.4 ABSTRACTION LAYERS

In the previous section, we investigated different strategies to control the random walk behavior aimed at capturing additional properties to local neighborhood. In this section, we investigate strategies that aim to include global structure through multiple layers of abstraction. These abstraction layers can be either encoded by representing the graph at multiple abstraction layers (and doing random walks on each layer) or imposed on the random walks itself.

We first present our approach to preserve the global graph structure by imposing hierarchical layers of abstraction on the random walks in the next section 12.4.1 on the facing page. We then highlight the key differences of our approach and related work to DeepWalk [147] in section 12.4.2 on the facing page. Finally, we evaluate the discussed methods in section 12.4.3 on page 170.

12.4.1 *Hierarchical Random Walk (HALK)*

Our approach, **H**ierarchical random w**ALK** (HALK), is applied on the sampled random walks. From those sampled walks, we remove a fraction of least frequent nodes at different levels. That is, we only keep the most frequent nodes in the walks, starting at a small fraction (e.g. 10%) and increase the retained fraction until we arrive back at the original walks. The intuition to train only the most frequent nodes in the beginning is to establish artificial connections between the most relevant nodes in the graph (hubs). With these artificial connections, we initialize the mapping in the embedding space by defining (close) relations between those hubs. That is, we represent the graph at different hierarchical abstraction layers, starting with the highest abstraction. The embeddings are then updated on more fine-grained layers by adding more and more nodes.

We initialize the feature representations of *all* nodes randomly, then we start training the representations of the *most frequent* nodes, using the first layer of reduced walks. At the next layer, we incorporate a larger fraction of nodes, update the already trained representations and train representations for the newly added nodes. We repeat this procedure, until we arrive back at the the original walks, i.e., we train representations for all nodes.

12.4.2 *Key Differences of HALK and Related Methods to DeepWalk*

Walklets [146] applies its modification after sampling the random walks, similar to our approach. In the sampled walks, nodes are skipped at different levels from 1 to s, where skip level 1 is equal to the original walk, i.e., not skipping nodes. Skipping nodes can be seen as adding artificial edges between nodes, e.g. skipping one node in the walk [a,b,c] would result in [a,c]. For each level, a separate model is trained, resulting in an overall dimensionality of the embeddings as dimensionality of a single model multiplied by the number of levels. These models are then combined into a single model with desired dimensionality (usually equal to the dimensionality of one of the previous single models) via PCA.

HARP [37] collapses the graph at different levels before sampling the walks. By collapsing, several adjacent nodes are aggregated and represented by a single node. Training the embeddings starts at the farthest collapsed graph, i.e., the most coarse graph, training only representations for nodes available at this level. The graph is populated back with more and more nodes on each level by un-collapsing the previously aggregated nodes, until walks are sampled from the original graph. During this procedure, representations of nodes from previous levels are updated, while those for newly added nodes in the current level are initially trained.

Our approach can be seen as a kind of combination between HARP and Walklets, as training on the most frequent nodes first can be seen as collapsing the graph to its hubs, similar to HARP. In terms of artificial connections, our approach is similar to Walklets, as we introduce them between hubs. Walklets adds artificial connections between every pair of nodes in the random walks, which occurs within s steps in the walk, where s is defined by the skip level.

12.4.3 *Evaluation*

We first reproduce evaluation results of DeepWalk [147], Walklets [146] and HARP [37] in order to guarantee a meaningful choice of parameters and correct implementation. As all three methods report results on the node classification task with BlogCatalog (cf. section 12.3.2 on page 164), we focus on this setup for reproduction in section 12.4.3.1 on the facing page. We then compare the three methods and ours on node classification with two additional datasets in section 12.4.3.2 on page 172.

Table 12.2 presents the basic statistics of the datasets used throughout the evaluation. Cora and Citeseer are citation networks, in which

Table 12.2: Datasets used in our experiments.

Dataset	#Vertices	#Edges	#Classes	Multi-Label?
Cora	2,708	5,429	7	no
BlogCatalog	10,312	333,983	39	yes
CiteSeer	3,312	4,732	6	no

the class indicates the research domain (single-label). BlogCatalog is a social network, in which edges represent friendship among bloggers and the classes represent topics a blogger is interested in (multi-label).

All methods build around random walks and the skip-gram model. If not mentioned otherwise, we use the following (common) parameter settings throughout the evaluation: Embedding dimension $d = 128$, sub-sampling threshold $t = 0.1$, context window size $m = 10$, negative samples $k = 5$ and five training iterations. The learning rate is decayed linearly per epoch, with an initial value of $\alpha = 0.025$ and a final value of $\alpha_{min} = 0.0001$. Rare nodes are not removed from the walks.

The following parameters are specific to the particular approach:

HARP has three options to collapse a set of adjacent nodes, edge-collapsing, star-collapsing and a hybrid between both. The second parameter specific to HARP is how far the graph is collapsed. By the default, the graph is collapsed to the furthest possible (i.e., only a single node would remain if collapsed further). We do not deviate from this default in any evaluation task.

HALK requires the number of layers and the percentage of most frequent nodes to keep at each layer to be defined. While the use of individual skip-gram training parameters per level would be possible, we only vary the amount of training iterations per layer and keep the remaining parameters the same throughout all levels.

WALKLETS skips nodes in a random walk. The number of nodes to skip is determined by the skip window size s (not to be confused with m, the context window size of the skip-gram model). Embeddings are trained for each skip level between 1 and s. Again individual skip-gram training parameters would be possible at each layer, but we keep them all the same.

12.4.3.1 *Reproduction of Results*

All compared methods report a score for node classification on BlogCatalog with a training fraction of 50%. Therefore, we selected this setup for reproduction. We tried to stick as close as possible to the original paper, by first collecting the parameter settings as described in the papers. If we could not find a parameter value in the paper, we tried to obtain it from the source code, if available. Otherwise we selected a reasonable value according to our experience.

The parameter settings for the individual approaches are as follows (ignoring common/unchanged parameters as described above).

DEEPWALK Perozzi et al. [147] report the number of walks as $n = 80$. The walk length is set to $l = 40$ (derived from source code).

WALKLETS The random walk parameters reported by the authors are $n = 1000$ and $l = 11$ [146]. The skip-level is set to $s = 2$, i.e., embeddings are trained on the original walks and additionally skipping one node. The embeddings of both levels are concatenated (resulting in $d = 256$) and PCA is used for dimensionality reduction in order to arrive back at $d = 128$.

HARP The random walk parameters are set to $n = 40$ and $l = 10$ as reported by the authors [37]. The graph collapsing scheme is the hybrid scheme of edge- and star-collapsing. On each collapsed graph, a single training iteration of the skip-gram model is performed, as the parameter was not reported by the authors and we did not observe a performance gain when increasing the training iterations. This behavior can be explained by HARP's graph coarsening: Training on several layers (24 for BlogCatalog) of the coarsened graphs effectively results in several iterations.

All authors report that they use a one-vs-rest logistic regression classifier with L2 regularization for their node classification task and we replicate this setup. The number of labels to predict is obtained

from an oracle. All methods report the Macro-F1 score, except for Walklets, reporting Micro-F1, which we follow in the reproduction. Each method uses a fraction of 50% for training. Scores are averaged over 10 random splits and we make sure that every method sees the same splits for training and test.

Table 12.3: Reproduction results, original score in right column. Macro-F1 reported for all methods except Walklets (Micro).

Method	Reproduction	Original
DeepWalk	27.84	27.30
HARP	24.84	24.66
Walklets	41.37	41.19

As clearly visible in table 12.3, our reproduced results are close to the reported ones and in particular the reproduced results are constantly better. The slight deviation can be explained by the random factor, both present in the initialization of embeddings and training/test splits. We can assume that our used parameters are mostly similar to those used in the original papers.

12.4.3.2 *Node Classification*

We evaluate HALK, HARP, DeepWalk and Walklets on the node classification task with the following datasets: Cora, BlogCatalog and CiteSeer (see table 12.2 on page 170 for basic statistics). First, we learn embeddings for all nodes in the graph and then we evaluate the same supervised classifier as in the reproduction experiment over training/test splits. For Cora and CiteSeer we use 90% of the data for training and 50% for BlogCatalog as BlogCatalog contains more vertices and edges. Similar to the reproduction experiment, we strictly ensure that all methods use the same data for training and testing to make a fair comparison possible. We report the Macro-F1 score (and standard deviation) averaged over 10 random splits.

As all compared methods are random walk based, we used the same parameters that determine the characteristics of the random walks for HARP, HALK, DeepWalk and Walklets. We also used the same representation and window size for these methods. Most parameters are similar to those that were already used in the reproduction of results. In particular, we set the number of walks $n = 80$, the walk length $l = 40$, window size $w = 10$, dimensions $d = 128$, negative samples $k = 5$ and sub-sampling threshold $t = 0.1$ for HALK, HARP, DeepWalk and Walklets. We also set the initial learning rate $\alpha = 0.025$, and linearly decay it to $\alpha_{min} = 0.001$ per epoch for all methods and keep rare nodes. For DeepWalk and Walklets, we set the number of iterations to 5. These settings mostly follow the original node

classification setup of DeepWalk. We set the skip window-size for Walklets to $s = 2$. We train HALK on 4 levels with 10%, 20%, 40% and 100% of the most frequent nodes and 10, 5, 3, 1 iterations respectively. While 10 iterations may seem rather high, one needs to consider that at this level, only 10% of the nodes are used for training, effectively resulting in less training time than a single iteration on the full data.

The results are presented in table 12.4. The best score per dataset

Table 12.4: Node classification Macro-F1 scores and standard deviation (±) over 10 random splits. Best score in bold, second-best underlined.

Algorithm/Dataset	Cora	BlogCatalog	CiteSeer
HALK	**81.65** ±2.2	**27.70** ±0.4	**56.97** ±3.2
HARP	<u>81.33</u> ±2.5	<u>27.65</u> ±0.6	<u>56.16</u> ±2.0
Walklets	81.10 ±2.5	27.31 ±0.5	55.76 ±2.9
DeepWalk	<u>81.33</u> ±2.0	27.57 ±0.5	55.54 ±2.8

is marked in bold, the second best is underlined. HALK performed best on all three datasets with our parameter settings. However, as differences in scores are all well within the standard deviation, there is practically no difference in performance. That means, we do not have a winner, but all methods perform equally well.

12.4.3.3 *Discussion*

As just mentioned, when considering the standard deviation, there is practically no difference in performance. Even more, running the evaluation a second time with the same parameter settings may result in a slightly different ranking. We observed this behavior for example for DeepWalk, creating several (DeepWalk) embedding models on the same set of random walks. Some of these models were then able to outperform the score of HALK on CiteSeer, but had a lower score than reported in the table on Cora. This behavior can be explained by the random initialization of the embeddings (as the randomness introduced by the random walks is eliminated by using the same set of walks).

All compared methods reported to outperform state of the art in their evaluation. However, this seems to be valid only for particular parameter choices. When it comes to absolute performance, all methods (including our own) share similar performance to DeepWalk and in particular cannot get above the almost 0.28 Macro-F1 score on BlogCatalog with a 50% training split as reported in the original presentation of DeepWalk [147]

12.5 CONCLUSION

In both, the evaluation of methods to control the random walk behavior and the evaluation of methods that train embeddings on multiple layers of the graph, we observe similar results: Extensions or modifications can improve performance under certain parameter settings. However, if the amount of random walks is sufficient and further hyper-parameters are well-chosen, these improvements vanish. Under these settings, we see a similar performance of all compared methods, in particular when it comes to ultimate performance. Neither the methods to control random walk behavior, nor the methods accounting for multiple abstraction layers of the graph were able to achieve a significantly higher Macro-F1 score than reported by Perozzi et al. for DeepWalk [147].

We therefore conclude that DeepWalk is still among the state of the art random walk based network embedding methods. In comparison, it exhibits less complexity and a small amount of hyper-parameters and is hence a valid choice for random walk based network embedding.

13 JOINT EMBEDDING OF NETWORK AND TEXT

In the previous chapter, we discussed methods that apply the skip-gram model with negative sampling to network embedding. Even before network embedding, the model, which was originally developed for word embeddings, has been extended to embed paragraphs or whole documents [108]. In this chapter, we discuss how to combine those two into a joint model for a unified representation of resources, regardless the information source (text or network). First, we explain the extension of the skip-gram model (Word2Vec) to documents (Doc2Vec). Then, we present our joint model and related approaches, which we evaluate and discuss afterwards before finally concluding the chapter.

We illustrate the discussion by the example of a paper citation network as depicted in figure 13.1. In this network, additional textual

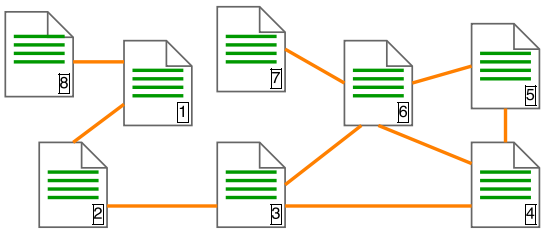


Figure 13.1: Paper citation network with **textual information** (document content) and **network structure** (citations).

information (document content) is associated with each node in the network. That is, each node represents a vertex in the network and a document at once.

13.1 FROM WORD TO DOCUMENT EMBEDDINGS

So far, we focused on Word2Vec's skip-gram model (SG) and in particular on the variant with negative sampling (SGNS) and only briefly mentioned the second architecture, continuous bag-of-words (CBOW) in section 10.2 on page 144. In this section, we provide more details on CBOW and then explain, how both model variants can be extend from word to document embeddings.

For illustration, we re-use the example from figure 10.1 on page 144, again illustrated in figure 13.2 on the following page. In this example, a sliding window is moved across the sentence "The quick brown fox jumped over the lazy dog", in order to define the context for a word. The sliding window in the example has a size of $m = 2$, i.e., two

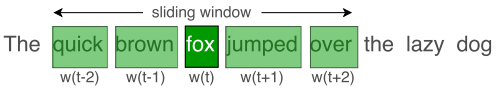
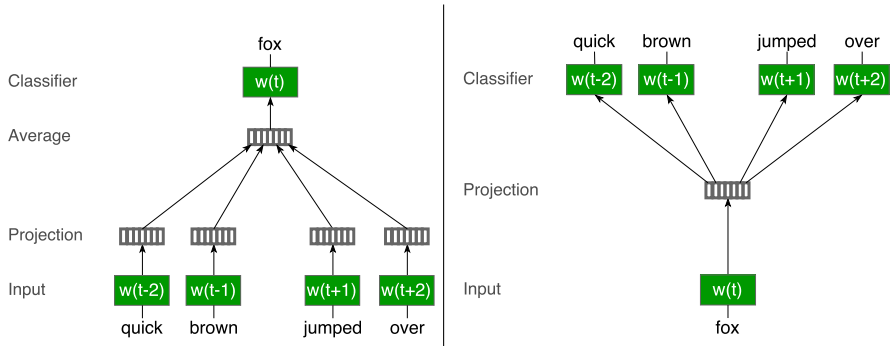


Figure 13.2: Illustration of word embedding sliding window with window size 2.

words to the left and two words to the right are considered context for the current center word. At the current window position, the context words for the center word "fox" are "quick", "brown", "jumped" and "over".

13.1.1 *Word Embeddings*

Mikolov et al. [132] proposed two model architectures to compute the embeddings: The continuous bag-of-words (CBOW) model, illustrated in figure 13.3a and the skip-gram (SG) model, illustrated in figure 13.3b. We discussed the latter already in section 10.2 on page 144. While



(a) continuous bag-of-words (CBOW)

(b) skip-gram (SG)

Figure 13.3: Illustration of the two model architectures of Word2Vec (CBOW and SG) with the sliding window context from the example sentence in figure 13.2. The projection corresponds to the weights between input and hidden layer, i.e., the word embeddings.

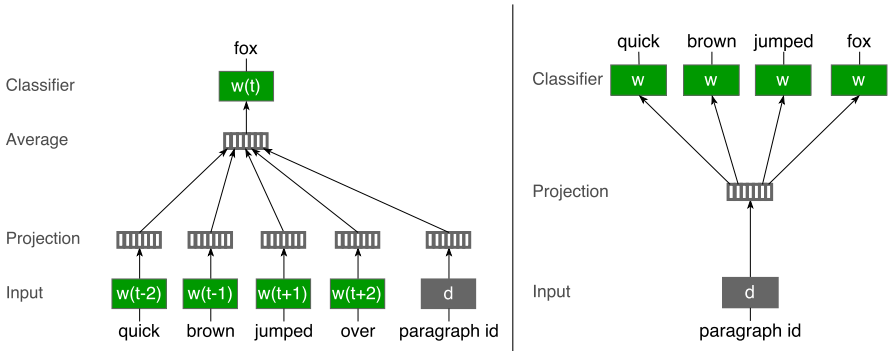
SG tries to predict the context, given the center word, CBOW tries to predict the center word, given the context.

Both models use a three-layer neural network, with linear activation in the hidden layer and a classifier in the output layer. In both models, this classifier can be either a softmax layer or negative sampling. The input in SG is a one-hot-coding and similarly in CBOW, a k-hot-coding activating the k inputs of the words in context. The output of the hidden layer in the CBOW model are the averaged weights between the activated inputs and the hidden layer. Similarly, in the SG model, only the weights from the input word are considered. No averaging is needed in this case and hence, the output of the hidden layer is just a copy of the weights between the active input and the hidden layer. The final embedding vector of a word in SG is given by the

weights between its corresponding input neuron and the hidden layer. Conversely in CBOW, the final embedding vector is defined by the weights between hidden layer and the output neuron corresponding to the center word.

13.1.2 *Document Embeddings*

Quoc and Mikolov extended the Word2Vec approach to sentences and documents with Paragraph Vector [108] (also known as Doc2Vec). Therefore, they extended the two architectures presented in the previous section with an additional paragraph identifier, yielding the distributed memory (PV-DM) model, illustrated in figure 13.4a and the distributed bag-of-words (PV-DBOW) model, illustrated in figure 13.4b.



(a) distributed memory (PV-DM)

(b) distributed bag-of-words (PV-DBOW)

Figure 13.4: Illustration of the two model architectures (PV-DM and PV-DBOW) of paragraph vector (Doc2Vec). The left part (PV-DM) shows the sliding window context from figure 13.2 on the facing page. PV-DBOW (right part) does not use a sliding window, instead all words from the sentence are considered as context.

While PV-DM extends the CBOW model, PV-DBOW corresponds to the SG model. In PV-DM, a sliding window is moved across the paragraph (or document), similar to sliding a window across the sentences in Word2Vec. In contrast to Word2Vec, an additional paragraph identifier is added to the context. This paragraph identifier can be thought of as another word, which acts as a memory that remembers what is missing from the current context – or the topic of the paragraph. Therefore, it got named distributed memory. PV-DBOW uses the paragraph identifier to predict the context words. Here, context words are the words contained in the paragraph. The model is trained by pairs of (<paragraph_id>, <word in paragraph>). In the figure, we illustrated only four context words, i.e., four training pairs. Of course, during training, all of the words in a paragraph are to be considered.

It is to note that the paragraph identifier is just a symbolic token, which carries no semantic meaning in itself. However, the embedding

of this identifier, i.e., the weights between the corresponding input neuron and the hidden layer (indicated as "projection" in the figure), does: After training the neural network, documents with similar content should have a similar representation in the embedding space, i.e., their weight vectors (projections) should be similar.

13.2 COMBINING LINK AND TEXT INFORMATION

In this section, we present a model to learn embeddings for documents and nodes jointly. We first show, how the Paragraph Vector approach as presented in section 13.1.2 on the previous page can be adapted to graph structures and then how to combine document with network embedding.

13.2.1 *Paragraph Vector on Graphs*

In DeepWalk [147], the Word2Vec [133] approach is adapted to learn node representations. As a pre-requisite for our model, we show how the Paragraph Vector approach can also be adapted to represent nodes. Therefore, we assign the start node of each random walk as the paragraph identifier and treat the rest of the walk as paragraph content. That means, we sample a rooted sub-graph around the starting node (root) and learn an embedding vector for the root of this sub-graph. If the walk length is chosen equal or larger than the network diameter, the random walks may cover the whole network. That is, the sampled sub-graph may not be a real sub-graph, but contain all nodes from the original graph.

This approach of sampling rooted sub-graphs is particularly relevant for PV-DM. PV-DBOW on the sampled sub-graphs could be seen as an equivalent to DeepWalk: If the walk length of PV-DBOW equals the window size of DeepWalk, both models have the same context radius. However, the distributions of sampled context pairs differ. Applying PV-DBOW on graphs in the aforementioned way, the amount of context pairs is the same for each starting node. On the contrary, in DeepWalk the amount of context pairs for hub nodes (i.e., nodes which are traversed by many random walks) is comparably higher. In preliminary experiments the performance of PV-DBOW on graphs was lower than DeepWalk, even when the amount of sampled random walks per node was set proportional to its degree. Therefore, we ignore PV-DBOW on graphs in the remainder and focus on DeepWalk.

13.2.2 *Fusing Link and Text Information*

We now have means to combine the models on both modalities, text and link information. For the combination, we simply share the paragraph (node) identifier between the document and the network model.

This is possible, as the paragraph (document) identifier is just a symbolic token and a document is a node in the network at the same time. Therefore, we can use the same identifier to refer to the node and document. To illustrate the joint model, we use the example sentence "The quick brown fox jumped over the lazy dog" from figure 13.2 on page 176 and the context window defined there. Assuming this sentence is the document content of node 2 in figure 13.1 on page 175, we also sample an exemplary walk starting at node 2 and illustrated in figure 13.5.

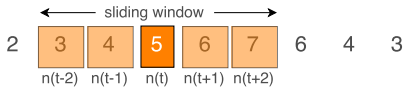


Figure 13.5: Illustration of random walk and sliding window with window size 2.

The combined model for the example sentence and walk is illustrated in figure 13.6. The illustrated model corresponds to the PV-DM approach, hence we refer to this architecture as text graph distributed memory (TG-DM). The left part of the figure is completely equivalent

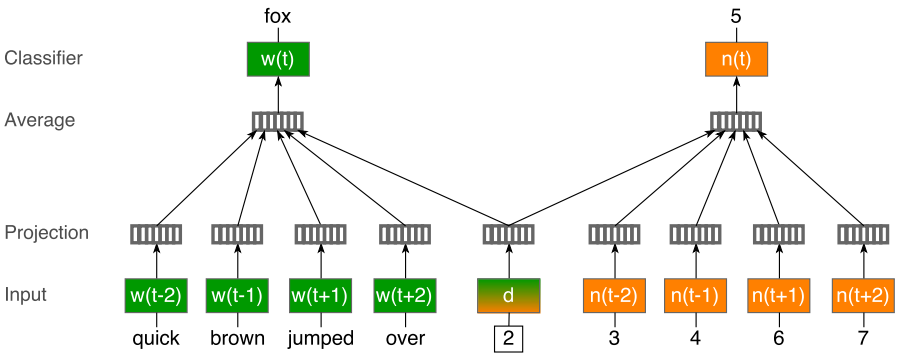


Figure 13.6: Illustration of text graph distributed memory (TG-DM) model. The paragraph vector d is shared between the document (left part) and the network model (right part), acting as distributed memory for both of them at the same time.

to PV-DM applied to textual content (cf. figure 13.4a on page 177), while the right part is equivalent to PV-DM for a sampled sub-graph (i.e., the set of walks with the same start node). These two models are combined via the shared weights of the paragraph identifier. This paragraph identifier is just a symbolic identifier, hence the "2" can be thought of as a document id and at the same time as a node id.

The combination of text and link information with the PV-DBOW approach is achieved in the exact same fashion, by sharing the paragraph identifier. Hence we refer to the combination via PV-DBOW as text graph distributed bag-of-words (TG-DBOW). The term *words* in this case also includes network nodes. As mentioned before, for the network part we apply DeepWalk instead of PV-DBOW on the graph.

Therefore, the textual context pairs are given by (<paragraph_id>, <word in paragraph>) and the network context pairs by (<center node>, <context node>) as defined by sliding the context window over the sampled walks. In this model, <paragraph_id> and <center node> refer to the same document/node with shared parameters and we denote it as paragraph identifier for simplicity.

We investigated different methods to train the TG-DBOW model, namely: TG-MIX, TG-SPLIT and TG-SUM, which we present in the following.

TG-DBOW uses the skip gram architecture and hence, the training consists of a paragraph identifier and context words and nodes. The simplest method is then to use the architecture of PV-DBOW and treat the context nodes from the sampled random walks as words in the document. We call this method TG-MIX, which is illustrated in figure 13.7.

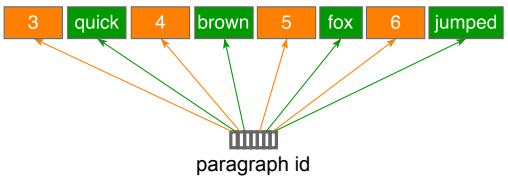


Figure 13.7: TG-MIX.

Another way to learn a combined embedding via TG-DBOW is to separate the output layers for nodes and words. We call this approach TG-SPLIT, illustrated in figure 13.8. In this case, we alternate between providing the model (paragraph identifier, context word) pairs and (paragraph identifier, context node) pairs. While the weights for the paragraph identifier are shared, the prediction of a context word or node is separated by using two separate output layers. These separate output layers are indicated with the two separate error measures E_1 and E_2 in the figure. Each output layer uses its own distribution of negative samples to draw from.

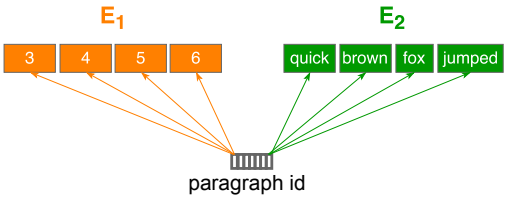


Figure 13.8: TG-SPLIT.

The third method, which we call TG-SUM, is illustrated in figure 13.9 on the facing page. Here, we train the network by providing triples of paragraph identifier, context word and context node. That is, for a given paragraph identifier, the network simultaneously has to predict the context word and the context node. In contrast to TG-SPLIT, where in each alternating step the error to be propagated back is either E_1 or

E_2, the error in TG-SUM is the sum of both errors (E_1 + E_2) in every step. That is, the weights of the paragraph identifier (embedding) are updated simultaneously by both modalities, text and link information.

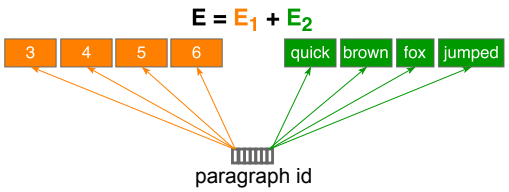


Figure 13.9: TG-SUM.

In preliminary experiments, TG-SPLIT outperformed TG-MIX and TG-SUM, hence we included only TG-SPLIT in the evaluation.

13.3 RELATED WORK

In this section, we present two closely related approaches (Paper2Vec and TADW) that also aim to combine text and link information. Further, we summarize related work on attributed network embedding, which has the focus on network embedding, but can also account for additional attributes. Recently, Zhang et al. [225] presented an evaluation of the same model as TG-SPLIT, which we will discuss in the evaluation section 13.4 on page 184.

13.3.1 *Paper2Vec*

With Paper2Vec, Ganguly and Pudi [56] do not propose a completely new model, but rather combine and extend existing models. Paper2Vec is modeled around a citation network, i.e., the data for this approach needs to have at least a similar structure (documents with textual content, that are connected via links). Their contribution to improve the learned representations is two-fold: First, they enrich the citation graph with artificial edges, based on the similarity of documents and second, they initialize the node embeddings with the paragraph vector representations obtained from the documents.

The intuition behind the enrichment of the citation graph with artificial edges is that those are links, which should actually be there, but are omitted because of the following potential reasons: Authors may not be fully aware of all the current work in the field and hence, might miss to cite some papers. Also, space constraints limit the amount of citable papers and trivially, two papers might appear at the same time and therefore the authors may not know about each others work. Adding artificial edges between similar documents can then be considered as adding the missing links. To this end, the authors first compute an embedding for each paper with the PV-DM approach.

Then, they add artificial links between each document and its k nearest neighbors, based on their similarity in the document embedding space.

To create the final embeddings, Paper2Vec applies the DeepWalk algorithm on the enriched graph that contains both, the original and artificial edges. Instead of randomly initializing the vector representations of the nodes in this graph, the authors initialize them with the document embeddings learned in the previous step. This raises the interesting question, whether this initialization avoids start configurations, that would result in a local optimum or whether the initialization preserves information from the document content, that cannot be learned via the network structure.

13.3.2 *Text-Associated DeepWalk (TADW)*

As a first important step of the TADW approach, Yang et al. [220] showed that DeepWalk factorizes a matrix of transition probabilities, similar to Word2Vec implicitly factorizing a word-context matrix [115]. More precisely, each entry in the matrix that gets factorized with DeepWalk is the logarithm of the average probability that vertex v_i randomly walks to vertex v_j in a fixed number of steps. Similarly, depending on the network architecture, an entry in the matrix that gets factorized by Word2Vec is the logarithm of the weighted co-occurrence between a (word, context word) pair [145] or the Shifted Positive Pointwise Mutual Information (SPMI) of the pair [115]. The matrix factorization of DeepWalk is illustrated in figure 13.10. The vertex representations of

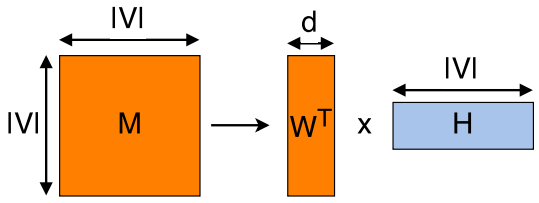


Figure 13.10: DeepWalk matrix factorization.

DeepWalk, i.e., the node embeddings, are given by the matrix W. The second step of TADW is then to incorporate text information. Instead of factorizing M into a product of two matrices, TADW factorizes M into a product of three matrices, where the third factor is a matrix of text features. This process is illustrated in figure 13.11 on the facing page. The text feature matrix T is a low-dimensional representation of the tf-idf matrix of the document contents, obtained via singular value decomposition. The matrices W and H are obtained by alternately optimizing them, subject to the following objective:

$$\min_{W,H} \left\| M - W^T HT \right\|_F^2 + \frac{\lambda}{2}(\|W\|_F^2 + \|H\|_F^2) \tag{13.1}$$

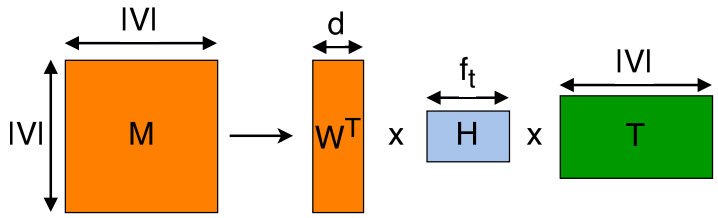


Figure 13.11: Text Associated DeepWalk (TADW) matrix factorization.

The final vector representation is obtained by concatenating W and $H \times T$, resulting in a $2 \cdot d$-dimensional embedding.

13.3.3 *Attributed Network Embedding*

Network embedding received increasing attraction within the recent years. With this increase, also more and more methods have been presented that incorporate additional information beyond the network structure, such as node content or link attributes. For example, Huang et al. [79] incorporate labels of nodes in learning the embedding. Kipf and Welling [96] also incorporate label information. Aiming to solve a node classification task, they focus on label information that is only partially available. Several authors [43, 77, 119] consider (categorical) node attributes as additional information source and the embedding should not only reflect node proximity within the network, but also proximity in terms of attributes. Zhang et al. [226] pursue a similar approach with emphasis on user profiles as node features. Li et al. [117] also account for both, network information and node attributes but put an explicit focus on dynamically changing networks (and attributes). Further work addressed the polarity of links, i.e., a link may be positive or negative, in particular in social networks [208, 209].

In most approaches for attributed network embedding (and also Paper2Vec), node or edge attributes serve as additional feature or input for the network embedding. The primary focus is still on the network structure. This is also evident in TADW: the final representation is a concatenation of W (which can be thought of as the network factor) and $H \times T$, where T is the text factor and H a kind of compatibility factor between W and T.

In contrast, our TG-SPLIT model values both modalities (text and link information) equally. Even more important, the model can cope with the absence of one modality in a particular node. That is, our model can be trained even when for some nodes only link or only text information is available.

13.4 EVALUATION

We evaluated the presented models (TG-DM and TG-SPLIT) on a node classification task on the CORA-ML dataset[1] and compared them against several baselines. This CORA-ML dataset contains 2708 scientific publications from the machine learning domain. Each paper is assigned to one of seven classes (e.g. neural networks or reinforcement learning). The citation network of these publications consists of 5429 links. Each document in the dataset is described as a binary vector of 1433 dimensions, indicating the presence of the corresponding word. The document contents are short texts, generated from title, containing 18 words on average. On this dataset, node classification is equivalent to document classification. A label is assigned to a node or document respectively, while both represent the same thing (a document is at the same time a node in the citation network).

For a quick reference, we list the approaches we compared again with a short description.

PV-DBOW (TEXT-ONLY) The distributed bag-of-words model of paragraph vector [108]. Document embeddings are trained by predicting the context, i.e., the words in the corresponding document.

PV-DM (TEXT-ONLY) The distributed memory model of paragraph vector [108]. Predicts a word from the context words together with a paragraph identifier. The paragraph identifier acts as a memory for the context and is the document embedding.

DW (LINK-ONLY) DeepWalk with the skip gram model [147]. Node embeddings are trained by predicting the context, i.e., the neighboring nodes on a sampled random walk.

CC (BOTH) Naive baseline concatenation of document (PV-DBOW) and node embeddings (DW).

TG-DM (BOTH) The paragraph vector model with distributed memory combining link and text information.

TG-SPLIT (BOTH) Our joint model of PV-DBOW and DeepWalk with the split approach, i.e., two separate output layers in the corresponding neural network.

P2V (BOTH) The Paper2Vec approach, adding artificial links and initializing node with document embeddings [56].

TADW (BOTH) Text associated DeepWalk, factorizing the matrix of transition probabilities into three matrices [220].

1 http://www.cs.umd.edu/~sen/lbc-proj/LBC.html – last accessed March 2020

We learned the vector representations on the whole dataset. Then we trained a support vector machine (SVM) on different portions of the data to predict the class in a supervised fashion, with the vector representations as input features. We opted for an SVM as it was also used in related work for the node classification task [56, 220]. The training portions were varied in 10% steps from 10% to 50%. We averaged the accuracy over 20 random splits for each percentage.

We explored different hyper-parameter settings with Bayesian Optimization [189], but did not find a major improvement. Therefore, we decided to stick to similar parameters as reported in the related work. We set the dimension of all the trained vector representations to $d = 100$. For the models including network information, we sampled $n = 10$ walks with a length of $t = 40$ for each node. We chose a window size of $m = 5$ and for models with negative sampling, we set the amount of samples to $k = 10$. As TG-SPLIT performed better than TG-SUM and TG-MIX in preliminary experiments, we only included TG-SPLIT in the evaluation.

13.4.0.1 *Results*

Beyond the results from our evaluation, table 13.1 on the next page also contains the results reported in the original papers of TADW and Paper2Vec for comparison. The column "source" indicates the paper, from which the results were copied and the corresponding rows are colored with a light gray background.

It is to note, that the document contents used for evaluation in Paper2Vec differ from the original CORA-ML dataset. In Paper2Vec, the authors crawled the full text of the documents, while the original dataset contains less information (18 words per document on average). This explains the rather large difference in the text-only baselines[2] and also has an influence on the other methods, that take the document content into account. Hence the results are not directly comparable. Further, the evaluation was carried out on random splits of the data (20 splits in our experiments, 10 splits in Paper2Vec and TADW). The actual splits are of course not the same, but the differences in the averaged results are less significant than the aforementioned differences in the data.

The lower accuracy of PV-DM compared to PV-DBOW (first two rows) is in line with related work, where PV-DBOW was observed to perform better on text classification tasks [217]. As just mentioned, the higher accuracy of text-only features reported by Paper2Vec is due to a different dataset (full-text vs. title). Accordingly, our result for the Paper2Vec approach with the original CORA-ML dataset is lower than reported in Paper2Vec. Interestingly the singular value decomposition

2 Even though the tf-idf dimensions are the same (1433) they contain different information. For tf-idf in Paper2Vec, the authors "kept the maximum features (sorted by df value) as 1433" [56].

Table 13.1: Detailed results, with averaged accuracy over random splits for every training percentage. Light gray rows indicate results reported in the paper corresponding to source. The use of full text documents in the evaluation data is indicated by *. Best values per group are marked bold for the original dataset, underlined for the full text dataset.

source	model (dimensions)	SVM training ratio 10%	20%	30%	40%	50%
	TEXT-ONLY					
	PV-DM (100)	43.3	48.9	51.8	53.7	54.9
	PV-DBOW (100)	57.9	62.3	64.4	65.7	66.7
p2v*	PV-DM (100)	76.8	81.1	82.8	83.5	84.4
	tf-idf (1433)	37.9	47.8	55.6	61.6	65.2
p2v*	tf-idf (1433)	78.5	82.8	84.7	86.0	86.9
tadw	tf-idf (200)	**58.3**	**67.4**	**71.1**	**73.3**	**74.0**
	LINK-ONLY					
	DW (100)	**77.2**	**80.7**	82.5	83.6	84.5
tadw	DW (100)	76.4	78.0	79.5	80.5	81.0
p2v*	DW (100)	76.0	**80.7**	**82.7**	**84.3**	**85.3**
	BOTH					
	P2V (100)	75.3	77.1	78.2	78.8	79.3
p2v*	P2V (100)	83.4	86.5	87.5	88.3	88.9
	TG-DM (100)	43.5	48.3	51.8	54.1	56.1
	TG-SPLIT (100)	73.4	79.7	82.3	83.9	84.9
	CC (200)	78.4	82.0	83.7	84.9	85.8
p2v*	CC (200)	80.6	83.8	85.1	86.4	87.1
tadw	CC (300)	76.5	80.4	82.3	83.3	84.1
p2v*	TADW (160)	82.4	85.0	85.6	86.0	86.7
tadw	TADW (200)	**82.4**	**85.0**	**85.6**	**86.0**	**86.7**

of tf-idf features performs better than tf-idf itself (and better than PV-DM and PV-DBOW).

The values reported by TADW and Paper2Vec and our evaluation for link-only embedding, using DeepWalk differ. These differences are in line with the observations from the previous chapter 12 on page 159.

The performance of TG-DM is by far lower than TG-SPLIT and the performance of the latter is slightly below the naive baseline of simply concatenating network and document embeddings (CC). TADW achieves the highest accuracy, outperforming the concatenation baseline. Strangely, Paper2Vec reports the exact same accuracy for TADW as reported by the original authors, even though different parameters and a different dataset were used in the evaluation of TADW performed by the authors of Paper2Vec (last two rows).

13.4.0.2 *Discussion*

With our model showing no improvement over the concatenated baseline, we assume that it is not able of making use of text information to complement link information and vice versa: Intuitively, we aim for similar embeddings of papers that have either similar content, similar citations or both. That is, papers with similar content should have a similar representation in the embedding space, regardless of their link structure (accordingly for papers with similar citations). This similarity criterion does not seem to be reflected in the embeddings, potentially because the optimization objective during learning the embeddings is to predict the context (or to predict from the context), not the class label.

Due to differences in the dataset, the accuracy score for Paper2Vec is lower than reported in the original paper. We attribute this to the lower performance of text features. Since those are used to add artificial links in Paper2Vec, the added links may be noisy, lowering the performance of node embeddings trained on the enriched network.

A potential reason for the better performance of TADW could be the higher accuracy in text features by the singular value decomposition of tf-idf features. If this higher accuracy of text features carries over to the TADW matrix factorization, it might be the beneficial factor for increased performance.

While our TG-SPLIT model does not improve over the naive baseline of simply concatenating document and network embeddings, it still has the advantage that it can be trained, even when text or link information is missing for some nodes. Using the same model, Zhang et al. [225] recently showed that even when text information is missing for 50% of the nodes, it has almost no influence on the model's performance (in terms of node classification accuracy). In contrast, the performance of other methods (such as e.g. TADW) drops drastically.

Part V

OUTLOOK

14 SUMMARY AND FUTURE WORK

In this final chapter, we briefly summarize the work in section 14.1 and point towards future research directions in section 14.2 on page 194

14.1 SUMMARY

This work addressed three research questions within the transformation of a user's information seeking process on the Web to zero-effort queries. The questions addressed the transformation from user generated queries to zero-effort queries, as well as the presentation of automatically retrieved results and the representation of resources. Each question is listed again subsequently and the results are briefly summarized.

RQ 1

How to transform the user's context into a query that retrieves relevant results?

We conducted a user study to gain a deep understanding how users search for related material and to collect ground truth data for developing zero-effort query models. The study revealed a high overlap between user generated queries and the current context (a text selection). This high overlap indicates that information on how to generate zero-effort queries is (partially) available from the context, even on syntactic level. Exploiting this overlap between context and user generated queries, we were able to identify the relevant terms with up to almost 90% accuracy. However, a text selection as explicit context indicator is rarely present in real world Web browsing and zero-effort queries that are similar to human queries are not necessarily *good queries* (i.e., retrieved results are relevant to the user). In fact, we empirically showed that a similar model suffers from low recall values when trying to learn *good queries* instead of *user queries*.

To remedy these issues, we developed means to automatic context detection on the more coarse grained level of a paragraph and zero-effort queries beyond a syntactic overlap. We presented a fast online method to extract paragraphs from a Web page without noticeable delay in typical usage scenarios and with a correctness above 80%. From those extracted paragraphs, we select the topmost paragraph in the viewport as focused as a simplistic solution, favoring transparency and user control. An eye-tracking study showed that detecting the

focus paragraph based on layout and interaction features could only significantly improve over this baseline for users with high mouse activity. Those users tend to "read with the mouse", but such behavior was only observable for 20% of the study participants. We developed a boolean query scheme that addresses the challenge of over- or under-specified queries for arbitrary search backends and consists of a main topic and additional keywords. The keywords are composed of named entities derived from the focus paragraph. Experimental results indicate the content of the whole page to be more suitable for deriving the main topic (instead of the focus paragraph). In a study utilizing that zero-effort query construction process, user modifications to the query could only slightly improve over the automatic queries, if at all. Based on the data collected in this study, we trained a model that outperformed the best user queries in terms of precision.

Personalization based on previous queries for which users viewed results showed beneficial in the same study. We address the cold start problem of personalization with an approach to bootstrap user profiles from social media, even for passive users. Passive users do not actively post content on social media, but express their interests by following other users or "liking" others' content. We showed that Twitter followee lists (the accounts a user follows) provide a sufficient basis for profile construction. High quality profiles can be mined by applying spreading activation to this information basis. A comparison of those passive profiles with active profiles showed high similarity on certain layers of abstraction, indicating a tight bond between consumption and production (in terms of interests).

RQ 2

How to adapt result presentation to zero-effort queries?

Presenting search results without explicit user interaction requires to adapt result presentation. We apply a ramping interface approach as proposed by Rhodes [157], gradually increasing the amount of information conveyed. From a study, eliciting user preferences for several notification and presentation styles and technical properties of Web browser add-ons, we derived guidelines for the implementation of zero-effort query result presentation interfaces.

Similarly, presenting results without explicit user interaction may leave the user unaware of how the retrieved results relate to the context or which information need the system assumes. In order to increase transparency, we display the generated query alongside with the retrieved results. Further, we developed a visual explanation that highlights the triadic relationship between context, query and related results, making connections explicit. Experimental results confirmed

an increase in users' efficiency when results are presented with this visual explanation.

Remembering where or how information was found in the past, i.e., recalling queries is already challenging in regular search, but even more when the search process is fully automatic. We presented QueryCrumbs, a simple-to-understand, compact visualization for accessing, altering and resubmitting previously issued queries. The understandability was verified in a formative user study, showing that interactions can be easily performed and visual encodings were well understood without instructions. The results indicate that QueryCrumbs can support users when searching for information in an iterative manner. A think-aloud test with search experts showed that these experts can gain valuable insights into the search engine backend by identifying specific patterns in the QueryCrumbs visualization. In a long-term usage study, we observed an uptake of the visualization, indicating that users deem QueryCrumbs beneficial for their search interactions.

RQ 3

How to represent resources to facilitate keyword-based search?

This question is not limited to our zero-effort query approach, but generally relevant to keyword-based retrieval on resource metadata. Resource metadata is typically represented in graph-like structures and accessed via rich and complex search interfaces. We search for a representation that allows a seamless integration of simple keyword queries and complex metadata. We focus on representation learning, i.e., learning feature representations by solving an optimization problem instead of hand-engineering. In particular, we focus on the skip-gram model with negative sampling (SGNS), applied to text and graph structures (by sampling random walks from the graph and treating them as sentence equivalents).

We optimized the convergence in terms of training epochs of SGNS, while retaining state-of-the art quality of the resulting word embeddings. Further, we investigated several extensions to SGNS applied on graphs proposed in related work and complemented them with our own modifications. The investigated extensions comprise means to control the random walk behavior and representing the graph on different layers of abstraction. The extensions only exhibited a performance gain over the baseline for particular parameter choices of the experimental setup. A gain in absolute performance was not observable, indicating no advantage of those extensions if the baseline parameters are chosen properly.

We presented a new model for embedding text and network data jointly. While experimental results showed no improvement over a

concatenation of separately trained embeddings in terms of node classification accuracy, our model can still be trained with partial absence of one modality.

14.2 FUTURE WORK

Enhancements and further research are possible in several aspects covered in this thesis. For example, in the zero-effort query generation process, we solely focused on textual context only, which could be extended to further media, such as images. Also the extraction and detection of the focus paragraph could be improved by taking content and its semantics into account. However, the last research question is the most important area for future work to us. Progress in this area is not limited to zero-effort queries, but beneficial to search in digital libraries in general. This thesis' research in terms of the third research question is only a first step towards a seamless integration of text-based retrieval and graph-like metadata representation.

In the future, we plan to improve the model developed in the last chapter to not only cope with the absence of text or network information during training, but also exploiting the information if both modalities are present in order to increase the model's performance. We aim to investigate integrating restrictions into the model, forcing it to learn embeddings that capture complementary information of each modality. Bridging textual representation and network metadata could also help to bridge the vocabulary gap between users and resource repositories, as search in those repositories is then no longer dependent on particular terms. We see further potential of such a bridge in the indexing of resources itself. Large scale multi-class/-label indexing is often still done manually [205], as automatic indexing poses a challenge, due to the high amount of classes, class imbalance and data sparsity [80]. Having only a few training examples available for certain classes requires special methods, such as few-shot learning [188]. Even beyond, zero-shot learning [142] addresses the challenge, when no training samples are available for a class at all, e.g. due to the emergence of new categories. We plan to investigate, how joint representations could be exploited in such few- and zero-shot scenarios.

So far, we discussed the representation of resources. In order to achieve the goal of a completely seamless integration of queries and resources, we also have to re-think the query representation. We pose it as an open question, whether it is sufficient to transform (user generated) keyword queries into the corresponding resource representation space or whether we need a fundamentally different query representation and if so, how such a representation could be realized in future information retrieval systems.

BIBLIOGRAPHY

[1] Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. "Analyzing user modeling on twitter for personalized news recommendations." In: *User Modeling, Adaption and Personalization*. Springer, 2011, pp. 1–12.

[2] Gediminas Adomavicius and Alexander Tuzhilin. "Context-Aware Recommender Systems." In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. Boston, MA: Springer US, 2011, pp. 217–253. ISBN: 978-0-387-85820-3. DOI: 10.1007/978-0-387-85820-3_7.

[3] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. "A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches." In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. NAACL '09. Boulder, Colorado: Association for Computational Linguistics, 2009, pp. 19–27. ISBN: 978-1-932432-41-1.

[4] Qingyao Ai, Liu Yang, Jiafeng Guo, and W. Bruce Croft. "Analysis of the Paragraph Vector Model for Information Retrieval." In: *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*. ICTIR '16. Newark, Delaware, USA: ACM, 2016, pp. 133–142. ISBN: 978-1-4503-4497-5. DOI: 10.1145/2970398.2970409.

[5] Cuneyt Gurcan Akcora, Barbara Carminati, Elena Ferrari, and Murat Kantarcioglu. "Detecting anomalies in social network data consumption." In: *Social Network Analysis and Mining* 4.1 (2014), pp. 1–16.

[6] James Allan, Bruce Croft, Alistair Moffat, and Mark Sanderson. "Frontiers, Challenges, and Opportunities for Information Retrieval: Report from SWIRL 2012." In: *SIGIR Forum* 46.1 (2012), pp. 2–32. ISSN: 0163-5840. DOI: 10.1145/2215676.2215678.

[7] Anne Aula, Rehan M. Khan, and Zhiwei Guan. "How Does Search Behavior Change As Search Becomes More Difficult?" In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10. Atlanta, Georgia, USA: ACM, 2010, pp. 35–44. ISBN: 978-1-60558-929-9. DOI: 10.1145/1753326.1753333.

[8] Juhee Bae, Vidya Setlur, and Benjamin Watson. "GraphTiles: A Visual Interface Supporting Browsing and Imprecise Mobile Search." In: *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*. MobileHCI '15. Copenhagen, Denmark: ACM, 2015, pp. 63–70. ISBN: 978-1-4503-3652-9. DOI: 10.1145/2785830.2785872.

[9] Seulki Bae and Youngmin Yi. "Acceleration of Word2vec Using GPUs." In: *Neural Information Processing*. Ed. by Akira Hirose, Seiichi Ozawa, Kenji Doya, Kazushi Ikeda, Minho Lee, and Derong Liu. Cham: Springer International Publishing, 2016, pp. 269–279. ISBN: 978-3-319-46672-9.

[10] Albert-László Barabási et al. *Network science*. Cambridge university press, 2016.

[11] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. "Gephi: an open source software for exploring and manipulating networks." In: *Third international AAAI conference on weblogs and social media*. 2009.

[12] Marcia J Bates. "Information search tactics." In: *Journal of the American Society for information Science* 30.4 (1979), pp. 205–214.

[13] Marcia J. Bates. "The Design of Browsing and Berrypicking Techniques for the Online Search Interface." In: *Online Review* 13.5 (1989), pp. 407–424.

[14] Benjamin B. Bederson. "Interfaces for Staying in the Flow." In: *Ubiquity* 2004.September (2004), pp. 1–1. ISSN: 1530-2180. DOI: 10.1145/1074068.1074069.

[15] Mikhail Belkin and Partha Niyogi. "Laplacian eigenmaps and spectral techniques for embedding and clustering." In: *Advances in neural information processing systems*. 2002, pp. 585–591.

[16] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives." In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

[17] Christoph Besel, Jörg Schlötterer, and Michael Granitzer. "Inferring Semantic Interest Profiles from Twitter Followees: Does Twitter Know Better than Your Friends?" In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. SAC '16. Pisa, Italy: Association for Computing Machinery, 2016, pp. 1152–1157. ISBN: 9781450337397. DOI: 10.1145/2851613.2851819.

[18] Christoph Besel, Jörg Schlötterer, and Michael Granitzer. "On the Quality of Semantic Interest Profiles for Onine Social Network Consumers." In: *SIGAPP Appl. Comput. Rev.* 16.3 (Nov. 2016), pp. 5–14. ISSN: 1559-6915. DOI: 10.1145/3015297.3015298.

[19] Krishna Bharat. "SearchPad: Explicit Capture of Search Context to Support Web Search." In: *Comput. Netw.* 33.1-6 (2000), pp. 493–501. ISSN: 1389-1286. DOI: 10.1016/S1389-1286(00)00047-5.

[20] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python*. O'Reilly Media, Inc., 2009.

[21] L Susan Blackford, Antoine Petitet, Roldan Pozo, Karin Remington, R Clint Whaley, James Demmel, Jack Dongarra, Iain Duff, Sven Hammarling, Greg Henry, et al. "An updated set of basic linear algebra subprograms (BLAS)." In: *ACM Transactions on Mathematical Software* 28.2 (2002), pp. 135–151.

[22] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. "Recommender systems survey." In: *Knowledge-Based Systems* 46 (2013), pp. 109–132. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2013.03.012.

[23] Daniel Boley, Maria Gini, Robert Gross, Eui-Hong(Sam) Han, Kyle Hastings, George Karypis, Vipin Kumar, Bamshad Mobasher, and Jerome Moore. "Document Categorization and Query Generation on the World Wide Web Using WebACE." English. In: *Artificial Intelligence Review* 13.5-6 (1999), pp. 365–391. ISSN: 0269-2821. DOI: 10.1023/A:1006592405320.

[24] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. "D3: Data-Driven Documents." In: *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2011).

[25] Cynthia A. Brewer, Geoffrey W. Hatchard, and Mark A. Harrower. "ColorBrewer in Print: A Catalog of Color Schemes for Maps." In: *Cartography and Geographic Information Science* 30.1 (2003), pp. 5–32. DOI: 10.1559/152304003100010929.

[26] Andrei Broder. "A Taxonomy of Web Search." In: *SIGIR Forum* 36.2 (2002), pp. 3–10. ISSN: 0163-5840. DOI: 10.1145/792550.792552.

[27] Harry Bruce, William Jones, and Susan Dumais. "Keeping and re-finding information on the web: What do people do and what do they need?" In: *Proceedings of the American Society for Information Science and Technology* 41.1 (2004), pp. 129–137. ISSN: 1550-8390. DOI: 10.1002/meet.1450410115.

[28] Jay Budzik and Kristian Hammond. "Watson: Anticipating and Contextualizing Information Needs." In: *62nd annaual meeting of the american society for information science*. 1999, pp. 727–740.

[29] J. J. Cadiz, Gina Venolia, Gavin Jancke, and Anoop Gupta. "Designing and Deploying an Information Awareness Interface." In: *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work*. CSCW '02. New Orleans, Louisiana,

USA: ACM, 2002, pp. 314–323. ISBN: 1-58113-560-2. DOI: 10.1145/587078.587122.

[30] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. "Extracting Content Structure for Web Pages Based on Visual Representation." In: *Proc. Asia-Pacific Web Conference*. Xian, China, 2003, pp. 406–417. ISBN: 3-540-02354-2.

[31] J. Canny, H. Zhao, B. Jaros, Y. Chen, and J. Mao. "Machine learning at the limit." In: *2015 IEEE International Conference on Big Data (Big Data)*. Oct. 2015, pp. 233–242. DOI: 10.1109/BigData.2015.7363760.

[32] Shaosheng Cao, Wei Lu, and Qiongkai Xu. "Grarep: Learning graph representations with global structural information." In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 2015, pp. 891–900.

[33] Shaosheng Cao, Wei Lu, and Qiongkai Xu. "Deep Neural Networks for Learning Graph Representations." In: *AAAI*. 2016, pp. 1145–1152.

[34] Robert Capra, Mary Pinney, and Perez-Quinones. *Refinding is Not Finding Again*. Tech. rep. TR-05-10. Computer Science, Virginia Tech., 2005.

[35] Stuart K. Card, George G. Robertson, and Jock D. Mackinlay. "The Information Visualizer, an Information Workspace." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '91. New Orleans, Louisiana, USA: ACM, 1991, pp. 181–186. ISBN: 0-89791-383-3. DOI: 10.1145/108844.108874.

[36] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian. "Grad-CAM++: Generalized Gradient-Based Visual Explanations for Deep Convolutional Networks." In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. Mar. 2018, pp. 839–847. DOI: 10.1109/WACV.2018.00097.

[37] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. "HARP: Hierarchical Representation Learning for Networks." In: *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*. 2018.

[38] Zhicong Cheng, Bin Gao, and Tie-Yan Liu. "Actively Predicting Diverse Search Intent from User Browsing Behaviors." In: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. Raleigh, North Carolina, USA: ACM, 2010, pp. 221–230. ISBN: 978-1-60558-799-8. DOI: 10.1145/1772690.1772714.

[39] Jeff Conklin. "Hypertext: An Introduction and Survey." In: *Computer* 20.9 (Sept. 1987), pp. 17–41. ISSN: 0018-9162. DOI: 10.1109/MC.1987.1663693.

[40] Lynn Sillipigni Connaway, Timothy J. Dickey, and Marie L. Radford. ""If it is too inconvenient I'm not going after it:" Convenience as a critical factor in information-seeking behaviors." In: *Library & Information Science Research* 33.3 (2011), pp. 179–190. ISSN: 0740-8188. DOI: https://doi.org/10.1016/j.lisr.2010.12.002.

[41] Edward Cutrell and Zhiwei Guan. "What Are You Looking for?: An Eye-tracking Study of Information Usage in Web Search." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '07. San Jose, California, USA: ACM, 2007, pp. 407–416. ISBN: 978-1-59593-593-9. DOI: 10.1145/1240624.1240690.

[42] Ali Dasdan, Paolo D'Alberto, Santanu Kolay, and Chris Drome. "Automatic Retrieval of Similar Content Using Search Engine Query Interface." In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. CIKM '09. Hong Kong, China: ACM, 2009, pp. 701–710. ISBN: 978-1-60558-512-3. DOI: 10.1145/1645953.1646043.

[43] Vachik S. Dave, Baichuan Zhang, Pin-Yu Chen, and Mohammad Al Hasan. "Neural-Brane: Neural Bayesian Personalized Ranking for Attributed Network Embedding." In: *Data Science and Engineering* 4.2 (June 2019), pp. 119–131. ISSN: 2364-1541. DOI: 10.1007/s41019-019-0092-x.

[44] Tangjian Deng and Ling Feng. "A survey on information re-finding techniques." In: *International Journal of Web Information Systems* 7 (4 2011), pp. 313–332. DOI: 10.1108/17440081111187538.

[45] Trien V. Do and Roy A. Ruddle. "The Design of a Visual History Tool to Help Users Refind Information Within a Website." In: *Proceedings of the 34th European Conference on Advances in Information Retrieval*. ECIR'12. Barcelona, Spain: Springer-Verlag, 2012, pp. 459–462. ISBN: 978-3-642-28996-5. DOI: 10.1007/978-3-642-28997-2_41.

[46] Mira Dontcheva, Steven M. Drucker, Geraldine Wade, David Salesin, and Michael F. Cohen. "Summarizing Personal Web Browsing Sessions." In: *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*. UIST '06. Montreux, Switzerland: ACM, 2006, pp. 115–124. ISBN: 1-59593-313-1. DOI: 10.1145/1166253.1166273.

[47] Paul Dourish. "What we talk about when we talk about context." In: *Personal Ubiquitous Comput.* 8.1 (2004), pp. 19–30. ISSN: 1617-4909. DOI: 10.1007/s00779-003-0253-8.

[48] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of Machine Learning Research* 12.Jul (2011), pp. 2121–2159.

[49] Angela Edmunds and Anne Morris. "The problem of information overload in business organisations: a review of the literature." In: *International Journal of Information Management* 20.1 (2000), pp. 17–28. ISSN: 0268-4012. DOI: http://dx.doi.org/10.1016/S0268-4012(99)00051-1.

[50] Carsten Eickhoff, Jaime Teevan, Ryen White, and Susan Dumais. "Lessons from the Journey: A Query Log Analysis of Within-session Learning." In: *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*. WSDM '14. New York, New York, USA: ACM, 2014, pp. 223–232. ISBN: 978-1-4503-2351-2. DOI: 10.1145/2556195.2556217.

[51] Stefano Faralli, Giovanni Stilo, and Paola Velardi. "Recommendation of microblog users based on hierarchical interest profiles." In: *Social Network Analysis and Mining* 5.1 (2015), pp. 1–23.

[52] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. "Placing search in context: The concept revisited." In: *ACM Transactions on information systems* 20.1 (2002), pp. 116–131. DOI: 10.1145/503104.503110.

[53] J. R. Firth. "A synopsis of linguistic theory 1930-55." In: *Studies in Linguistic Analysis (special volume of the Philological Society)*. Vol. 1952-59. Oxford: The Philological Society, 1957, pp. 1–32.

[54] Tiziano Flati, Daniele Vannella, Tommaso Pasini, and Roberto Navigli. "Two is bigger (and better) than one: the Wikipedia Bitaxonomy Project." In: *Proc. of ACL*. 2014, pp. 945–955.

[55] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. "A Survey on Concept Drift Adaptation." In: *ACM Comput. Surv.* 46.4 (2014), 44:1–44:37. ISSN: 0360-0300. DOI: 10.1145/2523813.

[56] Soumyajit Ganguly and Vikram Pudi. "Paper2vec: Combining Graph and Text Information for Scientific Paper Representation." In: *Advances in Information Retrieval: 39th European Conference on IR Research, ECIR 2017, Aberdeen, UK, April 8-13, 2017, Proceedings*. Cham: Springer International Publishing, 2017, pp. 383–395. ISBN: 978-3-319-56608-5.

[57] Fatih Gedikli, Dietmar Jannach, and Mouzhi Ge. "How should I explain? A comparison of different explanation types for recommender systems." In: *International Journal of Human-Computer*

Studies 72.4 (2014), pp. 367–382. ISSN: 1071-5819. DOI: https://doi.org/10.1016/j.ijhcs.2013.12.007.

[58] Jeremy Goecks and Jude Shavlik. "Learning Users' Interests by Unobtrusively Observing Their Normal Behavior." In: *Proc. IUI*. 2000.

[59] Tim Gollub, Matthias Hagen, Maximilian Michel, and Benno Stein. "From Keywords to Keyqueries: Content Descriptors for the Web." In: *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '13. Dublin, Ireland: ACM, 2013, pp. 981–984. ISBN: 978-1-4503-2034-4. DOI: 10.1145/2484028.2484181.

[60] Gene Golovchinsky, Morgan N Price, and Bill N Schilit. "From reading to retrieval: freeform ink annotations as queries." In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. Citeseer. 1999, pp. 19–25.

[61] Palash Goyal and Emilio Ferrara. "Graph Embedding Techniques, Applications, and Performance: A Survey." In: *arXiv preprint arXiv:1705.02801* (2017).

[62] Aditya Grover and Jure Leskovec. "Node2Vec: Scalable Feature Learning for Networks." In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, California, USA, 2016, pp. 855–864. ISBN: 978-1-4503-4232-2.

[63] Susan Gunelius. *Facebook's Growing Problem - Passive Users*. http://www.corporate-eye.com/main/facebooks-growing-problem-passive-users/. 2015.

[64] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. "Named Entity Recognition in Query." In: *Proc. of the 32nd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*. SIGIR '09. Boston, MA, USA: ACM, 2009, pp. 267–274. ISBN: 978-1-60558-483-6. DOI: 10.1145/1571941.1571989.

[65] Saurabh Gupta and Vineet Khare. "BlazingText: Scaling and Accelerating Word2Vec Using Multiple GPUs." In: *Proceedings of the Machine Learning on HPC Environments*. MLHPC'17. Denver, CO, USA: ACM, 2017, 6:1–6:5. ISBN: 978-1-4503-5137-9. DOI: 10.1145/3146347.3146354.

[66] Matthias Hagen, Anna Beyer, Tim Gollub, Kristof Komlossy, and Benno Stein. "Supporting Scholarly Search with Keyqueries." In: *Advances in Information Retrieval*. Ed. by Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Josiane Mothe, Fabrizio Silvestri, Giorgio Maria Di Nunzio, Claudia Hauff, and Gianmaria Silvello. Cham: Springer International Publishing, 2016, pp. 507–520. ISBN: 978-3-319-30671-1.

[67] Matthias Hagen, Martin Potthast, Anna Beyer, and Benno Stein. "Towards Optimum Query Segmentation: In Doubt Without." In: *Proc. of the 21st ACM Int. Conf. on Information and Knowledge Management*. CIKM '12. Maui, Hawaii, USA: ACM, 2012, pp. 1015–1024. ISBN: 978-1-4503-1156-4. DOI: 10.1145/2396761.2398398.

[68] Zellig Harris. "Distributional structure." In: *Word* 10.23 (1954), pp. 146–162.

[69] David Hauger, Alexandros Paramythis, and Stephan Weibelzahl. "Using Browser Interaction Data to Determine Page Reading Behavior." In: *Proc. UMAP*. 2011.

[70] Marti A. Hearst. "Multi-paragraph Segmentation of Expository Text." In: *Proc. Annual Meeting on Association for Computational Linguistics*. Las Cruces, New Mexico, 1994, pp. 9–16. DOI: 10.3115/981732.981734.

[71] Marti A. Hearst. *Search User Interfaces*. 1st. New York, NY, USA: Cambridge University Press, 2009. ISBN: 0521113792, 9780521113793.

[72] J. Heer, J. Mackinlay, C. Stolte, and M. Agrawala. "Graphical Histories for Visualization: Supporting Analysis, Communication, and Evaluation." In: *Visualization and Computer Graphics, IEEE Transactions on* 14.6 (Nov. 2008), pp. 1189–1196. ISSN: 1077-2626. DOI: 10.1109/TVCG.2008.137.

[73] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. "Generating Visual Explanations." In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Cham: Springer International Publishing, 2016, pp. 3–19. ISBN: 978-3-319-46493-0.

[74] I. Herman, G. Melancon, and M. S. Marshall. "Graph visualization and navigation in information visualization: A survey." In: *IEEE Transactions on Visualization and Computer Graphics* 6.1 (2000), pp. 24–43. DOI: 10.1109/2945.841119.

[75] Jeff Huang and Efthimis N. Efthimiadis. "Analyzing and Evaluating Query Reformulation Strategies in Web Search Logs." In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. CIKM '09. Hong Kong, China: ACM, 2009, pp. 77–86. ISBN: 978-1-60558-512-3. DOI: 10.1145/1645953.1645966.

[76] Jeff Huang, Ryen White, and Georg Buscher. "User See, User Point: Gaze and Cursor Alignment in Web Search." In: *Proc. Conference on Human Factors in Computing Systems*. SIGCHI. Austin, Texas, USA, 2012, pp. 1341–1350. ISBN: 978-1-4503-1015-4. DOI: 10.1145/2207676.2208591.

[77] Xiao Huang, Jundong Li, and Xia Hu. "Accelerated Attributed Network Embedding." In: *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 633–641. DOI: 10.1137/1.9781611974973.71.

[78] Xiao Huang, Jundong Li, and Xia Hu. "Label Informed Attributed Network Embedding." In: *WSDM '17*. 2017, pp. 731–739.

[79] Xiao Huang, Jundong Li, and Xia Hu. "Label Informed Attributed Network Embedding." In: *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. WSDM '17. Cambridge, United Kingdom: ACM, 2017, pp. 731–739. ISBN: 978-1-4503-4675-7. DOI: 10.1145/3018661.3018667.

[80] Himanshu Jain, Yashoteja Prabhu, and Manik Varma. "Extreme Multi-Label Loss Functions for Recommendation, Tagging, Ranking & Other Missing Label Applications." In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 935–944. ISBN: 9781450342322. DOI: 10.1145/2939672.2939756.

[81] Bernard J. Jansen, Amanda Spink, Chris Blakely, and Sherry Koshman. "Defining a Session on Web Search Engines: Research Articles." In: *J. Am. Soc. Inf. Sci. Technol.* 58.6 (2007), pp. 862–871. ISSN: 1532-2882. DOI: 10.1002/asi.v58:6.

[82] Bernard J. Jansen, Amanda Spink, and Jan Pedersen. "A Temporal Comparison of AltaVista Web Searching: Research Articles." In: *J. Am. Soc. Inf. Sci. Technol.* 56.6 (2005), pp. 559–570. ISSN: 1532-2882. DOI: 10.1002/asi.v56:6.

[83] Natalie Jhaveri and Kari-Jouko Räihä. "The advantages of a cross-session web workspace." In: *CHI '05 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '05. Portland, OR, USA: ACM, 2005, pp. 1949–1952. ISBN: 1-59593-002-7. DOI: 10.1145/1056808.1057064.

[84] S. Ji, N. Satish, S. Li, and P. K. Dubey. "Parallelizing Word2Vec in Shared and Distributed Memory." In: *IEEE Transactions on Parallel and Distributed Systems* 30.9 (Sept. 2019), pp. 2090–2100. ISSN: 2161-9883. DOI: 10.1109/TPDS.2019.2904058.

[85] Shihao Ji, Nadathur Satish, Sheng Li, and Pradeep Dubey. "Parallelizing Word2Vec in Multi-Core and Many-Core Architectures." In: *CoRR* abs/1611.06172 (2016).

[86] Jyun-Yu Jiang, Yen-Yu Ke, Pao-Yu Chien, and Pu-Jen Cheng. "Learning User Reformulation Behavior for Query Autocompletion." In: *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '14. Gold Coast, Queensland, Australia: ACM, 2014,

pp. 445–454. ISBN: 978-1-4503-2257-7. DOI: 10.1145/2600428.2609614.

[87] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. "Accurately Interpreting Clickthrough Data As Implicit Feedback." In: *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '05. Salvador, Brazil: ACM, 2005, pp. 154–161. ISBN: 1-59593-034-5. DOI: 10.1145/1076034.1076063.

[88] Ian T Jolliffe. "Principal Component Analysis and Factor Analysis." In: *Principal component analysis*. Springer, 1986, pp. 115–128.

[89] Sangaralingam Kajanan, Yang Bao, Anindya Datta, Debra VanderMeer, and Kaushik Dutta. "Efficient automatic search query formulation using phrase-level analysis." In: *Journal of the Association for Information Science and Technology* 65.5 (2014), pp. 1058–1075. DOI: 10.1002/asi.23022.

[90] Maryam Kamvar, Melanie Kellar, Rajan Patel, and Ya Xu. "Computers and Iphones and Mobile Phones, Oh My!: A Logs-based Comparison of Search Users on Different Devices." In: *Proceedings of the 18th International Conference on World Wide Web*. WWW '09. Madrid, Spain: ACM, 2009, pp. 801–810. ISBN: 978-1-60558-487-4. DOI: 10.1145/1526709.1526817.

[91] Pavan Kapanipathi, Prateek Jain, Chitra Venkataramani, and Amit Sheth. "User Interests Identification on Twitter Using a Hierarchical Knowledge Base." In: *The Semantic Web: Trends and Challenges*. Springer, 2014, pp. 99–113.

[92] Mohammad Mehdi Keikha, Maseud Rahgozar, and Masoud Asadpour. "Community aware random walk for network embedding." In: *Knowledge-Based Systems* 148 (2018), pp. 47–54. ISSN: 0950-7051.

[93] DanielA. Keim, Peter Bak, and Matthias Schäfer. "Dense Pixel Displays." English. In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M.TAMER ÖZSU. Springer US, 2009, pp. 789–795. ISBN: 978-0-387-35544-3. DOI: 10.1007/978-0-387-39940-9_1131.

[94] Megha Khosla, Jurek Leonhardt, Wolfgang Nejdl, and Avishek Anand. "Node Representation Learning for Directed Graphs." In: *arXiv preprint arXiv:1810.09176* (2018).

[95] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[96] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks." In: *International Conference on Learning Representations (ICLR)*. 2017.

[97] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks." In: *5th International Conference on Learning Representations (ICLR-17)*. 2017.

[98] Bart P. Knijnenburg and Martijn C. Willemsen. "Evaluating Recommender Systems with User Experiments." In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach, and Bracha Shapira. Boston, MA: Springer US, 2015, pp. 309–352. ISBN: 978-1-4899-7637-6. DOI: 10.1007/978-1-4899-7637-6_9.

[99] Christian Kohlschütter and Wolfgang Nejdl. "A Densitometric Approach to Web Page Segmentation." In: *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. CIKM '08. Napa Valley, California, USA: ACM, 2008, pp. 1173–1182. ISBN: 978-1-59593-991-3. DOI: 10.1145/1458082.1458237.

[100] Anita Komlodi, Gary Marchionini, and Dagobert Soergel. "Search History Support for Finding and Using Information: User Interface Design Recommendations from a User Study." In: *Inf. Process. Manage.* 43.1 (2007), pp. 10–29. ISSN: 0306-4573. DOI: 10.1016/j.future.2006.05.017.

[101] Anita Komlodi and Dagobert Soergel. "Search histories for user support in user interfaces." In: *J. Am. Soc. Inf. Sci.* 57 (2006), pp. 803–807. DOI: 10.1002/asi.20297.

[102] Joseph A. Konstan and John Riedl. "Recommender systems: from algorithms to user experience." In: *User Modeling and User-Adapted Interaction* 22.1 (Apr. 2012), pp. 101–123. ISSN: 1573-1391. DOI: 10.1007/s11257-011-9112-x.

[103] Robert Kreuzer, Jurriaan Hage, and Ad Feelders. "A Quantitative Comparison of Semantic Web Page Segmentation Approaches." In: *Engineering the Web in the Big Data Era*. Ed. by Philipp Cimiano, Flavius Frasincar, Geert-Jan Houben, and Daniel Schwabe. Cham: Springer International Publishing, 2015, pp. 374–391. ISBN: 978-3-319-19890-3.

[104] Joseph B Kruskal and Myron Wish. *Multidimensional scaling*. Vol. 11. Sage, 1978.

[105] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. "What is Twitter, a social network or a news media?" In: *WWW '10: Proceedings of the 19th international conference on World wide web*. Raleigh, North Carolina, USA: ACM, 2010, pp. 591–600. ISBN: 978-1-60558-799-8. DOI: http://doi.acm.org/10.1145/1772690.1772751.

[106] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data." In: *Proc. ICML*. 2001.

[107] Dmitry Lagun and Eugene Agichtein. "Inferring Searcher Attention by Jointly Modeling User Interactions and Content Salience." In: *Proc. Conference on Research and Development in Information Retrieval*. SIGIR '15. Santiago, Chile: ACM, 2015, pp. 483–492. ISBN: 978-1-4503-3621-5. DOI: 10.1145/2766462.2767745.

[108] Quoc Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents." In: *Proceedings of the 31st International Conference on Machine Learning*. Bejing, China: PMLR, 2014, pp. 1188–1196.

[109] Chia-Jung Lee and W. Bruce Croft. "Generating Queries from User-selected Text." In: *Proceedings of the 4th Information Interaction in Context Symposium*. IIIX '12. Nijmegen, The Netherlands: ACM, 2012, pp. 100–109. ISBN: 978-1-4503-1282-0. DOI: 10.1145/2362724.2362744.

[110] Kibeom Lee, Woon Seung Yeo, and Kyogu Lee. "Music Recommendation in the Personal Long Tail." In: *Proc. WOMRAD, ACM RecSys*. 2010.

[111] Ryong Lee and Kazutoshi Sumiya. "Zero-Effort Search and Integration Model for Augmented Web Applications." In: *Proc. of the 9th Int. Conf. on Web Engineering*. ICWE '9. San Sebastián, Spain: Springer-Verlag, 2009, pp. 330–339. ISBN: 978-3-642-02817-5. DOI: 10.1007/978-3-642-02818-2_27.

[112] S.L. LeMole, S.H. Nurenberg, J.T. O'Neil, and P.H. Stuntebeck. "Method and system for presenting customized advertising to a user on the world wide web." Patent US6009410 A (US). At&T Corporation. 1999.

[113] *Les Misérables network dataset – KONECT*. 2017.

[114] Mark Levene. *An Introduction to Search Engines and Web Navigation*. 2nd. 978-0-470-52684-2. Wiley, Sept. 2010.

[115] Omer Levy and Yoav Goldberg. "Neural Word Embedding as Implicit Matrix Factorization." In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pp. 2177–2185.

[116] Jane Li, Scott Huffman, and Akihito Tokuda. "Good Abandonment in Mobile and PC Internet Search." In: *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '09. Boston, MA, USA: ACM, 2009, pp. 43–50. ISBN: 978-1-60558-483-6. DOI: 10.1145/1571941.1571951.

[117] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. "Attributed Network Embedding for Learning in a Dynamic Environment." In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. CIKM '17. Singapore, Singapore: ACM, 2017, pp. 387–396. ISBN: 978-1-4503-4918-5. DOI: 10.1145/3132847.3132919.

[118] Xiaoli Li, Tong-Heng Phang, Minqing Hu, and Bing Liu. "Using Micro Information Units for Internet Search." In: *Proc. CIKM*. 2002.

[119] L. Liao, X. He, H. Zhang, and T. Chua. "Attributed Social Network Embedding." In: *IEEE Transactions on Knowledge and Data Engineering* 30.12 (Dec. 2018), pp. 2257–2270. ISSN: 2326-3865. DOI: 10.1109/TKDE.2018.2819980.

[120] Henry Lieberman. "Autonomous Interface Agents." In: *Proc. of the ACM SIGCHI Conf. on Human Factors in Compu. Sys.* CHI '97. Atlanta, Georgia, USA: ACM, 1997, pp. 67–74. ISBN: 0-89791-802-9. DOI: 10.1145/258549.258592.

[121] Henry Lieberman et al. "Letizia: An agent that assists web browsing." In: *IJCAI (1)* 1995 (1995), pp. 924–929.

[122] Kwan Hui Lim and Amitava Datta. "Interest classification of Twitter users using Wikipedia." In: *Proceedings of the 9th International Symposium on Open Collaboration*. ACM. 2013, p. 22.

[123] W. Liu, H. Li, and B. Xie. "Real-Time Graph Partition and Embedding of Large Network." In: *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. 2018, pp. 432–441.

[124] Chunliang Lu, Wai Lam, and Yingxiao Zhang. "Twitter user modeling and tweets recommendation based on wikipedia concept graph." In: *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*. 2012.

[125] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*. Vol. 1. Cambridge university press Cambridge, 2008.

[126] Gary Marchionini and Ryen White. "Find What You Need, Understand What You Find." In: *Int. J. Hum. Comput. Interaction* 23.3 (2007), pp. 205–237.

[127] Aleix M Martınez and Avinash C Kak. "Pca versus lda." In: *IEEE transactions on pattern analysis and machine intelligence* 23.2 (2001), pp. 228–233.

[128] Andrew Kachites McCallum. "MALLET: A Machine Learning for Language Toolkit." http://mallet.cs.umass.edu. 2002.

[129] Daniel C. McFarlane and Kara A. Latorella. "The Scope and Importance of Human Interruption in Human-computer Interaction Design." In: *Hum.-Comput. Interact.* 17.1 (2002), pp. 1–61. ISSN: 0737-0024. DOI: 10.1207/S15327051HCI1701_1.

[130] Sean M. McNee, John Riedl, and Joseph A. Konstan. "Being Accurate is Not Enough: How Accuracy Metrics Have Hurt Recommender Systems." In: *CHI '06 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '06. Montréal, Québec, Canada: ACM, 2006, pp. 1097–1101. ISBN: 1-59593-298-4. DOI: 10.1145/1125451.1125659.

[131] Rada Mihalcea and Paul Tarau. "TextRank: Bringing Order into Texts." In: *Conference on Empirical Methods in Natural Language Processing*. Barcelona, Spain, 2004.

[132] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space." In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. 2013.

[133] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[134] Dan Morris, Meredith Ringel Morris, and Gina Venolia. "SearchBar: A Search-centric Web History for Task Resumption and Information Re-finding." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '08. Florence, Italy: ACM, 2008, pp. 1207–1216. ISBN: 978-1-60558-011-1. DOI: 10.1145/1357054.1357242.

[135] Cun (Matthew) Mu, Guang Yang, and Yan (John) Zheng. "Revisiting Skip-Gram Negative Sampling Model with Rectification." In: *Intelligent Computing*. Ed. by Kohei Arai, Rahul Bhatia, and Supriya Kapoor. Cham: Springer International Publishing, 2019, pp. 485–497. ISBN: 978-3-030-22871-2.

[136] P. Mulhem and L. Nigay. "Interactive Information Retrieval Systems: From User Centered Interface Design to Software Design." In: *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '96. Zurich, Switzerland: ACM, 1996, pp. 326–334. ISBN: 0-89791-792-8. DOI: 10.1145/243199.243280.

[137] Cristiano Nascimento, Alberto H.F. Laender, Altigran S. da Silva, and Marcos André Gonçalves. "A Source Independent Framework for Research Paper Recommendation." In: *Proc. of the 11th JCDL*. 2011.

[138] Jakob Nielsen. *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. ISBN: 0125184050.

[139] Donald A. Norman. *The Design of Everyday Things*. Basic Books, 2002. ISBN: 0465067107.

[140] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. "Asymmetric Transitivity Preserving Graph Embedding." In: *KDD*. 2016, pp. 1105–1114.

[141] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. *The PageRank citation ranking: Bringing order to the web.* Tech. rep. Stanford InfoLab, 1999.

[142] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. "Zero-shot Learning with Semantic Output Codes." In: *Advances in Neural Information Processing Systems 22*. Ed. by Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta. Curran Associates, Inc., 2009, pp. 1410–1418.

[143] Yoon-Joo Park and Alexander Tuzhilin. "The long tail of recommender systems and how to leverage it." In: *Proc. of the RecSys '08*. 2008.

[144] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 8024–8035.

[145] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." In: *EMNLP*. 2014, pp. 1532–1543.

[146] Bryan Perozzi, Vivek Kulkarni, Haochen Chen, and Steven Skiena. "Don't Walk, Skip!: Online Learning of Multi-scale Network Embeddings." In: *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. 2017, pp. 258–265.

[147] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. "DeepWalk: Online Learning of Social Representations." In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '14. New York, New York, USA: ACM, 2014, pp. 701–710. ISBN: 978-1-4503-2956-9.

[148] Ravali Pochampally and Vasudeva Varma. "User context as a source of topic retrieval in Twitter." In: *Workshop on Enriching Information Retrieval (with ACM SIGIR)*. 2011, pp. 1–3.

[149] Pernilla Qvarfordt, Gene Golovchinsky, Tony Dunnigan, and Elena Agapie. "Looking Ahead: Query Preview in Exploratory Search." In: *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '13. Dublin, Ireland: ACM, 2013, pp. 243–252. ISBN: 978-1-4503-2034-4. DOI: 10.1145/2484028.2484084.

[150] Gabriëlle Ras, Marcel van Gerven, and Pim Haselager. "Explanation Methods in Deep Learning: Users, Values, Concerns and Challenges." In: *Explainable and Interpretable Models in Computer Vision and Machine Learning*. Ed. by Hugo Jair Escalante, Sergio Escalera, Isabelle Guyon, Xavier Baró, Yağmur Güçlütürk, Umut Güçlü, and Marcel van Gerven. Cham: Springer International Publishing, 2018, pp. 19–36. ISBN: 978-3-319-98131-4. DOI: 10.1007/978-3-319-98131-4_2.

[151] L. Ratinov, D. Roth, D. Downey, and M. Anderson. "Local and Global Algorithms for Disambiguation to Wikipedia." In: *ACL*. 2011.

[152] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. "Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent." In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger. Curran Associates, Inc., 2011, pp. 693–701.

[153] Radim Rehurek. *Word2vec in Python, Part Two: Optimizing*. https://rare-technologies.com/word2vec-in-python-part-two-optimizing/. Sept. 2013.

[154] Radim Rehurek and Petr Sojka. "Software Framework for Topic Modelling with Large Corpora." In: *IN PROCEEDINGS OF THE LREC 2010 WORKSHOP ON NEW CHALLENGES FOR NLP FRAMEWORKS*. 2010, pp. 45–50.

[155] Philip Resnik. "Using Information Content to Evaluate Semantic Similarity in a Taxonomy." In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*. IJCAI'95. Montreal, Quebec, Canada: Morgan Kaufmann Publishers Inc., 1995, pp. 448–453. ISBN: 1-55860-363-8, 978-1-558-60363-9.

[156] B. J. Rhodes and P. Maes. "Just-in-time Information Retrieval Agents." In: *IBM Syst. J.* 39.3-4 (2000), pp. 685–704. ISSN: 0018-8670. DOI: 10.1147/sj.393.0685.

[157] Bradley James Rhodes. "Just-In-Time Information Retrieval." PhD thesis. Massachusetts Institute of Technology, 2000.

[158] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. "Automatic Keyword Extraction from Individual Documents." In: *Text Mining*. John Wiley & Sons, Ltd, 2010, pp. 1–20. ISBN: 9780470689646. DOI: 10.1002/9780470689646.ch1.

[159] Sam T Roweis and Lawrence K Saul. "Nonlinear dimensionality reduction by locally linear embedding." In: *science* 290.5500 (2000), pp. 2323–2326.

[160] Tuukka Ruotsalo, Eetu Mäkelä, Tomi Kauppinen, Eero Hyvönen, Krister Haav, Ville Rantala, Matias Frosterus, Nima Dokoohaki, and Mihhail Matskin. "Smartmuseum: Personalized Context-aware Access to Digital Cultural Heritage." In: *Proc. ICSD*. 2009.

[161] Kunihiko Sakamoto. *Time to First Meaningful Paint*. https://goo.gl/vpaxv6. 2016.

[162] Andrés Sanoja and Stéphane Gançarski. "Web Page Segmentation Evaluation." In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. SAC '15. Salamanca, Spain: ACM, 2015, pp. 753–760. ISBN: 978-1-4503-3196-8. DOI: 10.1145/2695664.2695786.

[163] Fabian Schliski, Jörg Schlötterer, and Michael Granitzer. "Influence of Random Walk Parametrization on Graph Embeddings." In: *Advances in Information Retrieval*. ECIR 2020. to appear. 2020.

[164] Jörg Schlötterer. "From Context to Query." In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. SAC '15. Salamanca, Spain: Association for Computing Machinery, 2015, pp. 1108–1109. ISBN: 9781450331968. DOI: 10.1145/2695664.2696061.

[165] Jörg Schlötterer, Christin Seifert, and Michael Granitzer. "Web-based just-in-time retrieval for cultural content." In: *PATCH14: Proceedings of the 7th International ACM Workshop on Personalized Access to Cultural Heritage*. 2014.

[166] Jörg Schlötterer, Christin Seifert, and Michael Granitzer. "Supporting Web Surfers in Finding Related Material in Digital Library Repositories." In: *Research and Advanced Technology for Digital Libraries*. Ed. by Norbert Fuhr, László Kovács, Thomas Risse, and Wolfgang Nejdl. Cham: Springer International Publishing, 2016, pp. 434–437. ISBN: 978-3-319-43997-6.

[167] Jörg Schlötterer, Christin Seifert, and Michael Granitzer. "On Joint Representation Learning of Network Structure and Document Content." In: *Machine Learning and Knowledge Extraction*. Ed. by Andreas Holzinger, Peter Kieseberg, A Min Tjoa, and Edgar Weippl. Cham: Springer International Publishing, 2017, pp. 237–251. ISBN: 978-3-319-66808-6.

[168] Jörg Schlötterer, Christin Seifert, and Michael Granitzer. "QueryCrumbs for Experts: A Compact Visual Query Support System to Facilitate Insights into Search Engine Internals." In: *2018 22nd International Conference Information Visualisation (IV)*. July 2018, pp. 78–84. DOI: 10.1109/iV.2018.00024.

[169] Jörg Schlötterer, Christin Seifert, Wolfgang Lutz, and Michael Granitzer. "From Context-Aware to Context-Based: Mobile Just-In-Time Retrieval of Cultural Heritage Objects." In: *Advances in Information Retrieval*. Ed. by Allan Hanbury, Gabriella Kazai, Andreas Rauber, and Norbert Fuhr. ECIR 2015. Cham: Springer International Publishing, 2015, pp. 805–808. ISBN: 978-3-319-16354-3.

[170] Jörg Schlötterer, Christin Seifert, Christopher Satchell, and Michael Granitzer. "QueryCrumbs Search History Visualization - Usability, Transparency and Long-term Usage." In: *Journal of Computer Languages* (2020). to appear. DOI: https://doi.org/10.1016/j.cola.2020.100941.

[171] Jörg Schlötterer, Christin Seifert, Lisa Wagner, and Michael Granitzer. "A Game with a Purpose to Access Europe's Cultural Treasure." In: *GamifIR@ ECIR*. 2015, pp. 13–18.

[172] Jörg Schlötterer, Martin Wehking, Fatemeh Salehi Rizi, and Michael Granitzer. "Investigating Extensions to Random Walk Based Graph Embedding." In: *2019 IEEE International Conference on Cognitive Computing (ICCC)*. July 2019, pp. 81–89. DOI: 10.1109/ICCC.2019.00026.

[173] Christin Seifert, Werner Bailer, Thomas Orgel, Louis Gantner, Roman Kern, Hermann Ziak, Albin Petit, Jörg Schlötterer, Stefan Zwicklbauer, and Michael Granitzer. "Ubiquitous Access to Digital Cultural Heritage." In: *J. Comput. Cult. Herit.* 10.1 (Apr. 2017). ISSN: 1556-4673. DOI: 10.1145/3012284.

[174] Christin Seifert, Annett Mitschick, Jörg Schlötterer, and Raimund Dachselt. "Focus Paragraph Detection for Online Zero-Effort Queries: Lessons Learned from Eye-Tracking Data." In: *Proceedings of the 2017 Conference on Human Information Interaction and Retrieval*. CHIIR '17. Oslo, Norway: Association for Computing Machinery, 2017, pp. 301–304. ISBN: 9781450346771. DOI: 10.1145/3020165.3022138.

[175] Christin Seifert, Jörg Schlötterer, and Michael Granitzer. "Towards a Feature-rich Data Set for Personalized Access to Long-tail Content." In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. SAC '15. Salamanca, Spain: ACM, 2015, pp. 1031–1038. ISBN: 978-1-4503-3196-8. DOI: 10.1145/2695664.2695671.

[176] Christin Seifert, Jörg Schlötterer, and Michael Granitzer. "User Interface Considerations for Browser-Based Just-in-Time-Retrieval." In: *2015 19th International Conference on Information Visualisation*. July 2015, pp. 460–467. DOI: 10.1109/iV.2015.83.

[177] Christin Seifert, Jörg Schlötterer, and Michael Granitzer. "QueryCrumbs: A Compact Visualization for Navigating the Search Query History." In: *2017 21st International Conference Information Visualisation (IV)*. July 2017, pp. 35–44. DOI: 10.1109/iV.2017.23.

[178] Abigail J. Sellen, Rachel Murphy, and Kate L. Shaw. "How Knowledge Workers Use the Web." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '02. Minneapolis, Minnesota, USA: ACM, 2002, pp. 227–234. ISBN: 1-58113-453-3. DOI: 10.1145/503376.503418.

[179] Sara C. Sereno and Keith Rayner. "Measuring word recognition in reading: eye movements and event-related potentials." In: *Trends in Cognitive Sciences* 7.11 (2003), pp. 489–493. ISSN: 1364-6613. DOI: http://dx.doi.org/10.1016/j.tics.2003.09.010.

[180] J. Ševcech and M. Bieliková. "Query construction for related document search based on user annotations." In: *2013 Federated Conference on Computer Science and Information Systems*. Sept. 2013, pp. 279–286.

[181] Ben Shneiderman. "Promoting Universal Usability with Multi-layer Interface Design." In: *Proceedings of the 2003 Conference on Universal Usability*. CUU '03. Vancouver, British Columbia, Canada: ACM, 2003, pp. 1–8. ISBN: 1-58113-701-X. DOI: 10.1145/957205.957206.

[182] Ben Shneiderman, Don Byrd, and W Bruce Croft. "Clarifying search: a user-interface framework for text searches." In: *D-lib magazine* 3.1 (1997), pp. 18–20.

[183] Ben Shneiderman, Donald Byrd, and W. Bruce Croft. "Sorting out Searching: A User-interface Framework for Text Searches." In: *Commun. ACM* 41.4 (1998), pp. 95–98. ISSN: 0001-0782. DOI: 10.1145/273035.273069.

[184] Milad Shokouhi and Qi Guo. "From Queries to Cards: Re-ranking Proactive Card Recommendations Based on Reactive Search History." In: *Proc. of the 38th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*. SIGIR '15. Santiago, Chile: ACM, 2015, pp. 695–704. ISBN: 978-1-4503-3621-5. DOI: 10.1145/2766462.2767705.

[185] Patrick Siehndel and Ricardo Kawase. "TwikiMe!: User Profiles That Make Sense." In: *Posters and Demonstrations Track*. ISWC-PD'12. Boston, USA: CEUR-WS.org, 2012, pp. 61–64.

[186] Rashmi Sinha and Kirsten Swearingen. "The Role of Transparency in Recommender Systems." In: *CHI '02 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '02. Minneapolis, Minnesota, USA: ACM, 2002, pp. 830–831. ISBN: 1-58113-454-1. DOI: 10.1145/506443.506619.

[187] Alastair G. Smith. "Internet search tactics." In: *Online Information Review* 36.1 (2012), pp. 7–20. DOI: 10.1108/14684521211219481.

[188] Jake Snell, Kevin Swersky, and Richard Zemel. "Prototypical Networks for Few-shot Learning." In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 4077–4087.

[189] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. "Practical Bayesian Optimization of Machine Learning Algorithms." In: *Proceedings of the 25th International Conference on Neural Information Processing Systems*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 2951–2959.

[190] Daniel Sondheim, Geoffrey Rockwell, Mihaela Ilovan, Milena Radzikowska, and Stan Ruecker. "Interfacing the Collection." In: *Scholarly and Research Communication* 3.1 (2012).

[191] Caroline Sporleder and Mirella Lapata. "Automatic Paragraph Identification: A Study across Languages and Domains." In: *Proc. Conference on Empirical Methods in NLP*. EMNLP. 2004, pp. 72–79.

[192] Sven Strickroth and Niels Pinkwart. "High quality recommendations for small communities: the case of a regional parent network." In: *Proc. Recsys*. 2012.

[193] Alistair Sutcliffe and Mark Ennis. "Towards a cognitive theory of information retrieval." In: *Interacting with Computers* 10.3 (1998), pp. 321–351. ISSN: 0953-5438. DOI: http://dx.doi.org/10.1016/S0953-5438(98)00013-7.

[194] Kirsten Swearingen and Rashmi Sinha. "Interaction Design for Recommender Systems." In: *In Designing Interactive Systems 2002. ACM*. Press, 2002.

[195] Mona Taghavi, Ahmed Patel, Nikita Schmidt, Christopher Wills, and Yiqi Tew. "An Analysis of Web Proxy Logs with Query Distribution Pattern Approach for Search Engines." In: *Comput. Stand. Interfaces* 34.1 (2012), pp. 162–170. ISSN: 0920-5489. DOI: 10.1016/j.csi.2011.07.001.

[196] Lynda Tamine-Lechani, Mohand Boughanem, and Nesrine Zemirli. "Inferring the user interests using the search history." In: *Workshop on information retrieval, Learning, Knowledge and Adaptatbility (LWA 2006)*. 2006, pp. 108–110.

[197] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN: 0321321367.

[198] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. "LINE: Large-scale Information Network Embedding." In: *Proceedings of the 24th International Conference on World Wide Web*. 2015, pp. 1067–1077.

[199] Ke Tao, Fabian Abel, Qi Gao, and Geert-Jan Houben. "Tums: twitter-based user modeling service." In: *The Semantic Web: ESWC 2011 Workshops*. Springer. 2012, pp. 269–283.

[200] Tobii Technology. *Accuracy and precision Test report TX300*. Tech. rep. Tobii Technology AB, 2013.

[201] Jaime Teevan. "The Re:Search Engine: Simultaneous Support for Finding and Re-finding." In: *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*. UIST '07. Newport, Rhode Island, USA: ACM, 2007, pp. 23–32. ISBN: 978-1-59593-679-0. DOI: 10.1145/1294211.1294217.

[202] Joshua B Tenenbaum, Vin De Silva, and John C Langford. "A global geometric framework for nonlinear dimensionality reduction." In: *science* 290.5500 (2000), pp. 2319–2323.

[203] N. Tintarev and J. Masthoff. "A Survey of Explanations in Recommender Systems." In: *2007 IEEE 23rd International Conference on Data Engineering Workshop*. Apr. 2007, pp. 801–810. DOI: 10.1109/ICDEW.2007.4401070.

[204] Nava Tintarev and Judith Masthoff. "Evaluating the effectiveness of explanations for recommender systems." In: *User Modeling and User-Adapted Interaction* 22.4 (Oct. 2012), pp. 399–439. ISSN: 1573-1391. DOI: 10.1007/s11257-011-9117-5.

[205] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al. "An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition." In: *BMC bioinformatics* 16.1 (2015), p. 138.

[206] JBP Vuurens, C Eickhoff, AP de Vries, and MF Balcan. "Efficient Parallel Learning of Word2Vec." In: *Balcan, MF (ed.), ICML'16: The 33rd International Conference on Machine Learning, June 24, New York, 2016. ML Systems Workshop*. New York: JMLR. 2016, pp. 1–4.

[207] Daixin Wang, Peng Cui, and Wenwu Zhu. "Structural deep network embedding." In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016, pp. 1225–1234.

[208] Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. "SHINE: Signed Heterogeneous Information Network Embedding for Sentiment Link Prediction." In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. WSDM '18. Marina Del Rey, CA, USA: ACM, 2018, pp. 592–600. ISBN: 978-1-4503-5581-0. DOI: 10.1145/3159652.3159666.

[209] Suhang Wang, Charu Aggarwal, Jiliang Tang, and Huan Liu. "Attributed Signed Network Embedding." In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. CIKM '17. Singapore, Singapore: ACM, 2017, pp. 137–146. ISBN: 978-1-4503-4918-5. DOI: 10.1145/3132847.3132905.

[210] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. "Community Preserving Network Embedding." In: *AAAI' 17*. 2017, pp. 203–209.

[211] Yiwen Wang, Lora M. Aroyo, Natalia Stash, and Lloyd Rutledge. "Interactive User Modeling for Personalized Access to Museum Collections: The Rijksmuseum Case Study." In: *User Modeling*. Springer Berlin Heidelberg, 2007.

[212] Colin Ware. *Information Visualization: Perception for Design*. 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2012. ISBN: 9780123814647, 9780123814654.

[213] Harald Weinreich, Hartmut Obendorf, Eelco Herder, and Matthias Mayer. "Off the Beaten Tracks: Exploring Three Aspects of Web Navigation." In: *Proceedings of the 15th International Conference on World Wide Web*. WWW '06. Edinburgh, Scotland: ACM, 2006, pp. 133–142. ISBN: 1-59593-323-9. DOI: 10.1145/1135777.1135802.

[214] Ryen W. White and Steven M. Drucker. "Investigating Behavioral Variability in Web Search." In: *Proceedings of the 16th International Conference on World Wide Web*. WWW '07. Banff, Alberta, Canada: ACM, 2007, pp. 21–30. ISBN: 978-1-59593-654-7. DOI: 10.1145/1242572.1242576.

[215] Ryen W. White, Susan T. Dumais, and Jaime Teevan. "Characterizing the Influence of Domain Expertise on Web Search Behavior." In: *Proceedings of the Second ACM International Conference on Web Search and Data Mining*. WSDM '09. Barcelona, Spain: ACM, 2009, pp. 132–141. ISBN: 978-1-60558-390-7. DOI: 10.1145/1498759.1498819.

[216] Ryen W. White and Dan Morris. "Investigating the Querying and Browsing Behavior of Advanced Search Engine Users." In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR

'07. Amsterdam, The Netherlands: ACM, 2007, pp. 255–262. ISBN: 978-1-59593-597-7. DOI: 10.1145/1277741.1277787.

[217] Nils Witt and Christin Seifert. "Understanding the Influence of Hyperparameters on Text Embeddings for Text Classification Tasks." In: *Research and Advanced Technology for Digital Libraries*. Ed. by Jaap Kamps, Giannis Tsakonas, Yannis Manolopoulos, Lazaros Iliadis, and Ioannis Karydis. Cham: Springer International Publishing, 2017, pp. 193–204. ISBN: 978-3-319-67008-9.

[218] Seiji Yamada, Naoki Mori, and Kazuki Kobayashi. "Peripheral Agent: Implementation of Peripheral Cognition Technology." In: *CHI '13 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '13. Paris, France: ACM, 2013, pp. 1701–1706. ISBN: 978-1-4503-1952-2. DOI: 10.1145/2468356.2468661.

[219] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Qiang Yang, and Stephen Lin. "Graph embedding and extensions: A general framework for dimensionality reduction." In: *IEEE transactions on pattern analysis and machine intelligence* 29.1 (2007), pp. 40–51.

[220] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. "Network Representation Learning with Rich Text Information." In: *Proceedings of the 24th International Conference on Artificial Intelligence*. 2015, pp. 2111–2117.

[221] Cheng Yang, Maosong Sun, Zhiyuan Liu, and Cunchao Tu. "Fast network embedding enhancement via high order proximity approximation." In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*. 2017, pp. 19–25.

[222] Yin Yang, Nilesh Bansal, Wisam Dakka, Panagiotis Ipeirotis, Nick Koudas, and Dimitris Papadias. "Query by Document." In: *Proceedings of the Second ACM International Conference on Web Search and Data Mining*. 2009.

[223] Wu Youyou, Michal Kosinski, and David Stillwell. "Computer-based personality judgments are more accurate than those made by humans." In: *Proceedings of the National Academy of Sciences* 112.4 (2015), pp. 1036–1040.

[224] R. Zafarani and H. Liu. *Social Computing Data Repository at ASU*. 2009.

[225] D. Zhang, J. Yin, X. Zhu, and C. Zhang. "SINE: Scalable Incomplete Network Embedding." In: *2018 IEEE International Conference on Data Mining (ICDM)*. Nov. 2018, pp. 737–746. DOI: 10.1109/ICDM.2018.00089.

[226] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. "User profile preserving social network embedding." In: *IJCAI International Joint Conference on Artificial Intelligence*. 2017.

[227] Yi Zhang, Jianguo Lu, and Ofer Shai. "Improve Network Embeddings with Regularization." In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. CIKM '18. Torino, Italy: ACM, 2018, pp. 1643–1646. ISBN: 978-1-4503-6014-2. DOI: 10.1145/3269206.3269320.

[228] Yi Zhang, Fen Zhao, and Jianguo Lu. "P2V: large-scale academic paper embedding." In: *Scientometrics* 121.1 (Oct. 2019), pp. 399–432. ISSN: 1588-2861. DOI: 10.1007/s11192-019-03206-9.

[229] Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. "DoSeR - A Knowledge-Base-Agnostic Framework for Entity Disambiguation Using Semantic Embeddings." In: *The Semantic Web. Latest Advances and New Domains*. Ed. by Harald Sack, Eva Blomqvist, Mathieu d'Aquin, Chiara Ghidini, Simone Paolo Ponzetto, and Christoph Lange. Cham: Springer International Publishing, 2016, pp. 182–198. ISBN: 978-3-319-34129-3.

[230] Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. "Robust and Collective Entity Disambiguation through Semantic Embeddings." In: *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '16. Pisa, Italy: Association for Computing Machinery, 2016, pp. 425–434. ISBN: 9781450340694. DOI: 10.1145/2911451.2911535.