Technische Universität Berlin

A|O|T
Agententechnologien in betrieblichen Anwendungen und der Telekommunikation

Development and Evaluation of a Large-scale Recommender System for Wikipedia

Master Thesis
am Fachgebiet Agententechnologien in betrieblichen Anwendungen und der Telekommunikation (AOT)
Prof. Dr.-Ing. habil. Sahin Albayrak
Fakultät IV Elektrotechnik und Informatik
Technische Universität Berlin

vorgelegt von
Malte Schwarzer

Betreuer:	Dr.-Ing. Andreas Lommatzsch
Gutachter:	Prof. Dr. Dr. h.c. Sahin Albayrak
	Prof. Dr. habil. Odej Kao

Malte Schwarzer
Matrikelnummer: 340819

Erklärung der Urheberschaft

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Ort, Datum Unterschrift

Abstract

Recommender systems (RS) support users in filtering the vast and increasing number of documents on the Web. Text-based methods for recommending documents are well-established among websites. In particular, the MoreLikeThis (MLT) component from the Apache Lucence framework is a common basis for text-based RS. Wikipedia, for instance, is currently using a MLT-based component to recommend articles to its users. A recent study shows that the link-based Co-Citation Proximity Analysis (CPA) is a promising alternative to MLT. The study compares MLT and CPA for the task of recommending Wikipedia articles, even if CPA was originally developed to find related scientific papers based on citations.

With this thesis, we continue the research on MLT- and CPA-based RS for Wikipedia, since the previous study could not give a final statement on which method is superior. To answer this question, we develop CPA from a research prototype into a large-scale RS, which can be deployed into the production environment of Wikipedia. Such deployment gives the opportunity of a large-scale online evaluation with millions of users of the Wikipedia Android app. We demonstrate the validity of this evaluation approach with a lab study. The feedback from 33 participants shows no statistical significant difference between CPA and MLT. Both RS perform similarly well with respect to the click-oriented evaluation. The outcome of the user study corresponds to the outcome of a manual analysis of sample recommendation sets, which also does not reveal any significant difference of CPA and MLT. Thus, we consider the online evaluation as representative despite the low number of participants. A real-world study with Wikipedia's production system could not be conducted as part of this thesis.

Prior to the user-centric evaluation, we implement and evaluate conceptional improvements of CPA. The introduction of an Inverse Link Frequency factor increases the click rate, precision and coverage of CPA's recommendations. Furthermore, the definition of co-link proximity as number of words between links outperforms other proximity definitions. We tested the improvements in an offline evaluation based on quasi-gold standards from Wikipedia's "See also" links and click streams.

The thesis shows that a CPA-based RS is suitable for a large-scale deployment in a production environment. Moreover, our findings confirm CPA's promising results from the preliminary study. CPA's performance is at least equivalent to MLT's, whereby in future work the superiority of one method needs to be determined with a real-world online evaluation.

Zusammenfassung

Empfehlungsdienste unterstützen Benutzer beim Bewältigen der im Web allgegenwärtigen Informationsflut. Textbasierte Verfahren zählen dabei zu den gängigsten Empfehlungsdiensten. Insbesondere die MoreLikeThis (MLT) Komponente aus dem Apache Lucene Framework ist ein bei vielen Webdiensten beliebter Empfehlungsdienst. Beispielsweise benutzt Wikipedia MLT für Artikelempfehlungen. Eine aktuelle Studie zeigt, dass das linkbasierte Verfahren Co-Citation Proximity Analysis (CPA), welches ursprünglich für das zitationsbasierte Empfehlen von Wissenschaftsliteratur entwickelt wurde, eine vielversprechende Alternative zu MLT sein kann. Diese Studie hat CPA und MLT an Hand von Wikipedia mit einer Offline-Auswertung verglichen.

Die hier vorliegende Arbeit setzt die Forschung über CPA und MLT als Wikipedia-Empfehlungsdienst fort, denn die vorherige Studie konnte nicht feststellen, welcher Empfehlungsdienst besser ist. Um diese Frage zu beantworten, wird CPA von einem Prototypen zu einem in dem Produktivsystem der Wikipedia einsetzbaren Empfehlungsdienst entwickelt. Ein solcher Empfehlungsdienst ermöglicht die großangelegten Online-Auswertung mit der Wikipedia Android-App. Das entwickelte System wird im Rahmen einer Laborstudie mit 33 Teilnehmern evaluiert. Die dabei ermittelten Ergebnisse zeigen, dass beide Verfahren ähnlich gute Ergebnisse liefern. Hinsichtlich der Ergebnisgüte konnte kein signifikanter Unterschied zwischen den Verfahren CPA und MLT festgestellt werden. Eine Studie mit dem Produktivsystem der Wikipedia konnte im Rahmen dieser Arbeit nicht durchgeführt werden.

Vor der Durchführung der Laborstudie wurden konzeptionelle Verbesserung von CPA entwickelt und quantitativ mit “See also”-Links und Clickstreams ausgewertet. Ein “Inverse Link Frequency”-Faktor verbessert die Klickraten, Genauigkeit und Empfehlungsvielfalt von CPA. Des Weiteren konnte gezeigt werden, dass die Anzahl von Wörtern zwischen zwei Links am Besten die “Co-Link Proximity” bestimmt.

Zusammengefasst zeigt diese Forschungsarbeit, dass CPA für den großangelegten Einsatz als Empfehlungsdienst in einem Produktivsystem geeignet ist. Außerdem konnten die Ergebnisse der vorherigen Studie bestätigt werden. CPA erzielt mindestens mit MLT vergleichbare Ergebnisse, wobei eine abschließende Beurteilung mit einer großangelegten Online-Auswertung erfolgen sollte.

Acknowledgements

I would like to thank all the participants of my user study, DIMA for generously allowing me to use the IBM power cluster, the Wikimedia Foundation, especially everybody from the ticket system and chat, who provided answers to all the questions that I had regarding MediaWiki and the Wikipedia setup.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we explain the motivation behind this thesis and introduce our research goals and how we plan to achieve them.

1.1 Motivation

Recommender systems are crucial filtering and discovery tools to manage the vast and continuously increasing volume of items available in digital libraries and on the Web. The research community usually distinguished two types of recommender systems: Content-based and collaborative filtering. Collaborative filtering requires user data and is often used by Web services like social networks that are able to collect and analyze large amount of user behavior information. However, most recommender system (approximately 55%) employ content-based document features and corresponding similarity measures to provide recommendations [6].

Especially in academia, recommender systems are a centerpiece among research support tools. Keeping track of the latest research in one's field by identifying the most relevant papers is essential for research progress. The exponentially increasing number of published articles and the increased speed of article availability, makes thorough literature research even more important, but at the same time more tedious and time consuming for researchers. In academic recommender systems, citation-based features and document similarity measures have proven valuable [30, 44], while in the Web-context they are less common.

Wikipedia is an other example for a large and rapidly growing digital library. As of April 2016, all language-specific versions of the Wikipedia combined contain

approximately 39 million articles, of which five million are in English[1]. The English Wikipedia grew by approximately 1,000 articles per day in 2015. All Wikimedia projects received on average 18 billion page views (crawlers excluded) per month in 2015[2]. Despite Wikipedia's size, popularity and rapid growth, little research has addressed the issue of improving information search in Wikipedia through automated generation of article recommendations. At the same time, Wikipedia neglects the collection of user behavior data due to privacy concerns and, therefore, collaborative filtering is ineligible. Hence, we aim to further investigate the effectiveness of content-based Wikipedia recommender systems. In particular, the comparison of text- and link-based concepts is the matter of this thesis.

1.2 Objective

In an initial study [39], Co-Citation Proximity Analysis (CPA) [1] has shown promising results for the task of recommending related Wikipedia articles. With this thesis we continue the research on this topic.

The initial study compared CPA with classical Co-Citation Analysis (CoCit) and MoreLikeThis (MLT) from the Apache Lucence framework[3]. MLT is a text-based approach and relies on the concept of Term Frequency - Inverse Document Frequency (TF-IDF) [36]. While being widely used by other web services, MLT is also currently in use by Wikipedia itself. In [39], the three recommender systems were compared in an offline evaluation, which did not allow a final statement on the superiority of one method. Primary, the study was only data-centric, i.e. no real users were involved. With this thesis we aim to overcome this drawback by performing an online evaluation with real user feedback.

To be able to perform an online evaluation, we need to develop and deploy a test framework into the Wikipedia infrastructure that is capable of distributing article recommendations and measuring how user interact with the recommendations. The test framework should be primarily used to compare MLT and CPA in a real-world scenario. The recommendations generated by CPA should be optimized for the integration into the Wikipedia Android app[4], where implicit user feedback, i.e. clicks on the recommendations, can be collected. The online evaluation can be conducted

[1] `http://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia`
[2] `http://reportcard.wmflabs.org`
[3] `http://lucene.apache.org/`
[4] `https://play.google.com/store/apps/details?id=org.wikipedia`

in either a large-scale experiment, where the recommendations will be exposed to the general Wikipedia audience, or in a small-scale lab study, where users are explicitly ask to participate in the experiment.

Prior to the online evaluation, we test possible improvements of CPA, which the preliminary study [39] proposed as future work. The effectiveness of the improvements are evaluated in an offline experiment. Moreover, the thesis investigates how the technical implementation needs to be adjusted to deploy CPA in Wikipedia's production environment. A productive system such as the Wikipedia infrastructure presents strict requirements that need to be met by the recommender system. The in the previous study developed research prototype of the CPA recommender system does not meet these requirements.

All in all, the thesis boils down to the following research goals:

1. Which approach, link-based Co-Citation Proximity Analysis (CPA) or text-based MoreLikeThis (MLT), performs better in an online evaluation for recommending related Wikipedia articles?

2. Given the simplicity of the current CPA approach, how can additional modifications improve CPA's recommendation performance?

3. Can CPA be implemented in such a way that it can be deployed in a large-scale productive environment like the Wikipedia infrastructure?

The approach to answer to the research questions, the corresponding work and the findings itself are presented in the following chapters.

1.3 Structure of the Thesis

This thesis is structured as follows: In Chapter 2, we discuss essential background information related to the thesis. We introduce Wikipedia in the context of this thesis, the technologies, like Apache Flink or Elasticsearch, that were used in this research, and the algorithms behind the evaluated recommender systems. Chapter 3 gives an overview on similar recommender system studies and other related literature. In Chapter 4, we present the details of the technical implementation. In particular, we describe the system architecture and its components. Evaluation results are presented in Chapter 5. Chapter 6 summarizes the findings, gives conclusion to the research objectives and proposes areas of future work.

Chapter 2

Background

In this chapter, we describe at first the research problem. Then, we introduce technologies, data and concepts that we use throughout this work and we motivate why they were used.

2.1 Problem Description

With respect the research objectives, we describe in the following the problem that we solve with this work, focusing on three main aspects: Relevance judgment of recommendations, the evaluation method and Wikipedia's implementations requirements.

2.1.1 Relevance Judgment

The general goal of recommender systems is to recommend items that are relevant to users. In our case we want to recommend Wikipedia articles that are relevant in context of the currently read article to users of the Wikipedia Android app. However, determining relevance is a complex challenge due to its nature. Relevance consists of two main components [22]:

- Commonly objective topical relevance
- Purely subjective user relevance

Domain experts can judge the topical relevance of a document. On contrary, user relevance highly depends on the user's information need. In the latter case,

relevance of items can differ from user to user. As a result the automatic relevance judgment is challenging. Moreover, recommender systems need to balance both relevance components. Collaborative filtering systems usually individualize recommendations. However, current content-based recommender systems identify topically relevant documents by using document similarity as approximation of relevance [23]. Here, the idea is that documents, which are similar, are more likely to cover a related topic. Yet, pure similarity neglects objective and subjective relevance of a document. This leads to the problem of relevance is not being reflected in content-based recommendations.

This problem also applies on the Wikipedia use case. Therefore, we focus on objective topical relevance when recommending articles, since only the article content and no user data is taken into account and, therefore, no subjective user relevance can be expected.

2.1.2 Evaluation Method

Recommender system researchers face one major difficulty: The performance evaluation of recommender systems, i.e. researchers need to prove the advantage of their approach. Reproducibility and significance are thereby issues. Even if there are various evaluation metrics available to measure the recommendation performance, the evaluation methodology is the crucial part. Throughout the literature, there are three types of evaluation methods widely established [18, 41]: (1) Offline evaluations, where recommender systems are compared without user interaction and based on an existing data set. (2) Online evaluations, which involve the collection of novel user signals in an recommender system application. (3) User-studies, where a group of subjects test the recommender system in an lab or real-world scenario.

Offline Evaluation

Offline evaluations make use of existing data and can be conducted without any real user interaction. For this reason offline evaluations are also called data- or system-centric evaluations, whereas online and user-studies are called user-centric. A common approach is to a take predefined recommendation set and the corresponding user feedback, remove user feedback from the set and measure how well a recommender system can predict or recover the missing feedback. The user feedback can be an explicit rating or another action like a view or a sale. In the latter case it is not clear if a user action implies that the user likes the recommendation. The user, for

instance, can buy a product and afterwards rate it negatively or cancel the order. In offline evaluations the performance is usually measured with an accuracy metrics, e.g. precision and recall. Due to the missing user involvement, offline evaluations usually require the least effort of the three types. Thus, an offline evaluation should be used to test a recommender system in an early prototyping-stage. However, the outcome of an offline evaluation strongly depends on the underlying data set, which might include bias. An offline evaluation alone cannot provide enough evidence to prove that one system outperforms the other. Hence, it is recommended to always combine an offline evaluation with at least one of the other two types. Offline evaluations are also often used in recommendation challenges, e.g. The Million Song Dataset Challenge [29] or Netflix Price [9], as different implementations by different authors can be compared by this method.

User Study

In the user study setting, the evaluation involves user interaction, i.e. real user provides feedback on the recommendations. In a qualitative study explicit feedback is provided, typically with a questionnaire or interview. In this case the subject is aware of him or her taking part in the study. One problem is that it is questionable whether the results can be transferred to the true user satisfaction, since the group of subjects can be skewed: A user might act different if he or she knows about the experiment. Only users with the need of sharing their opinion participate in a questionnaire. A significant large number of well-chosen subjects can overcome this problem. Yet, conducting a user study at this scale is often unfeasible due to limited resources. Also, finding participants for a niche domains can be tedious. Despite this challenge, user studies can provide quality and in-depth feedback on the performance of recommender system, that cannot be measured by any other evaluation method. Unifying evaluation frameworks like ResQue [33] and LensKit [14] exist to standardize user studies.

Online Evaluation

The barrier for conducting an online evaluations is significantly higher compared to the other evaluation methods. That is because an online evaluation requires that the researchers own or have access to a deployed recommender system. The application, where the recommender system is deployed, e.g. a website, should as well have a large enough number of active users and should allow to track how its users are interacting

with the recommender system. With such a system implicit user feedback can be collected, e.g. by counting the clicks on specific recommendations. An explicit user action is not required for the evaluation. Therefore, an online evaluation allows to gather large amount of user feedback. Making the outcome of such a study more meaningful and less biased, as long as the researchers ensure that the feedback is not affect by other factors, e.g. seasonal trends or other modification of the system.

Wikipedia offers an elegant solution to the evaluation method problem. For an offline evaluation Wikipedia provides quasi-gold standards that can be used as data set to test recommender systems. Finding participants for a Wikipedia user-study is also comparably easy, since most Internet users are already familiar with Wikipedia. But most interestingly, Wikipedia's open source policy also gives the opportunity to research to contribute their research work and to potentially evaluate it with the Wikipedia community.

2.1.3 Implementation Requirements

Research prototypes of recommender systems usually neglect runtime and stability issues. This holds true for the in [39] developed CPA recommender system. With respect to the goal of conducting an online evaluation and deploying the recommender system to Wikipedia's production infrastructure, the CPA prototype needs to be adjusted to fit the requirement of such deployment.

First of all, the CPA recommender system needs to be integrable into Wikipedia's infrastructure. Put differently, we must rely on all existing technologies and applications and introduce as little as possible new to Wikipedia. Otherwise Wikipedia's developer would not accept a novel feature because of incalculable maintenance overhead. This is a restriction at most open source software projects.

Furthermore, resource management is crucial for a non-profit organization such as Wikipedia. Due to resource limitation, the recommender system needs to be optimized with respect to its computing time. At the same time, the recommender system needs to be capable of generating recommendations for a Wikipedia languages in frequent time intervals, e.g. once a week, to reflect to content changes.

2.2 Wikipedia

From the research perspective Wikipedia is interesting as test collection because of its diversity and size (Section 1.1). In the following, we give some background information on the two tested Wikipedia languages, German and simple English, the corresponding data sets and quasi-gold standards that we extract from Wikipedia. To clarify the terminology, we distinguish the Wikipedia website (*wikipedia.org* and its language versions), the Wikipedia Android app, the Wikipedia articles (test collection or corpus) and the Wikimedia Foundation (WMF), which is the organization that runs all services related to the Wikipedia website. With Wikipedia languages, e.g. German Wikipedia, we refer to the articles available on the German version of the Wikipedia website (*de.wikipedia.org*). When using the terms Wikipedia infrastructure or production environment, we refer to the technical infrastructure, e.g. Web and database servers, that hosts the Wikipedia website.

2.2.1 Wikipedia as Test Collection

In the preliminary study [39], we investigated the English Wikipedia. Yet, for this thesis we want to extend the work on other languages to test whether the language or the corpus size affects the performance. For this reason and because of limited computing resources, which prevent the use in our user study, we do not consider the English Wikipedia in this thesis. Instead, we investigate German and simple English[1] Wikipedias as research subject.

Generally a link-based recommender system as CPA should be language independent in contrast to text-based approaches. Following this idea you would assume that the recommendation algorithm performs similar well on all Wikipedia versions. However, the link usage within the corpus strongly affects the performance from CPA as shown in the previous study. Hence, the different size of the German and simple English test collection is more important than the different language of the words. Both test collections are smaller than the English Wikipedia, which is the biggest Wikipedia language with five million articles. With 2,015,558 million articles German is the second biggest Wikipedia language, whereas simple English is relatively small with 122,076 articles. All numbers are based on the Wikipedia XML dump

[1]Simple English usually refers to a simplified form of the English language, such as: Basic English, a controlled language, created by Charles Kay Ogden, which only contains a small number of words.

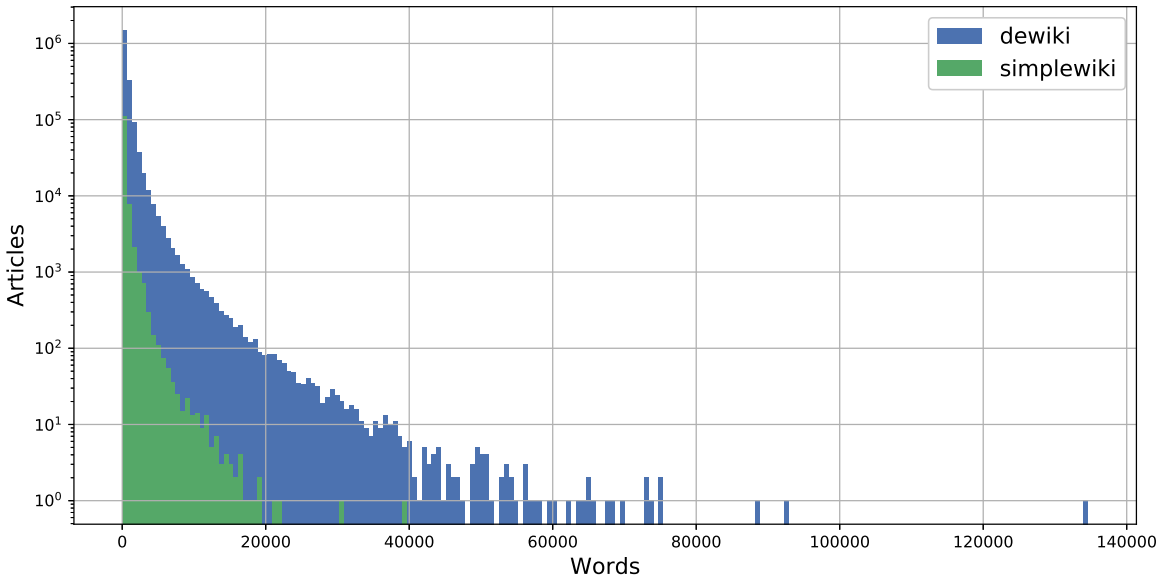

Figure 2.1: Distribution of number of words per Wikipedia article (log scale). Very long articles are rare. German articles are generally longer than simple English articles.

from January 2017[2].

Many articles of the simple English Wikipedia are shorter than the same articles in the English Wikipedia [43]. On average simple English articles have 337 words, while the longest article has 39,452 words. German articles are generally longer (avg. length 657 words, max. 134,621 words). Most articles are short, while only a few very long article exist (Figure 2.1). 94% articles in the simple English Wikipedia have less than 1,000 words (85% in German).

Looking at in-links per Wikipedia article (Figure 2.2), the distribution is even more skewed. 95% articles in the simple English Wikipedia have less than 50 in-links (93% in German). While simple English articles receive on average 12.2 in-links and at maximum 12,657 in-links (German: avg. 23.6 in-links, max. 121,340 in-links).

Consequently, both recommender systems have the challenge to deal with articles that have very different properties, whereby the word count should have a stronger

[2] `https://dumps.wikimedia.org/`

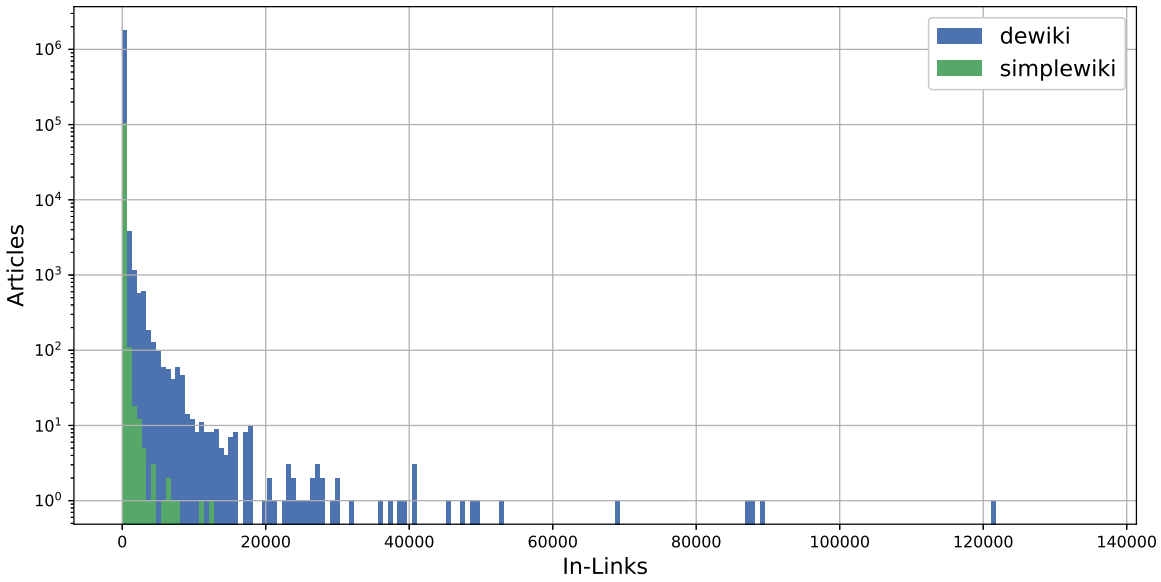

Figure 2.2: Distribution of number of in-links per Wikipedia article (log scale). Most articles have none or only a few in-links.

effect on MLT and the in-link count on CPA. Regarding the languages, we expect the recommender system to perform better on German, since it generally has article with more words and in-links.

2.2.2 Wikipedia as Quasi-gold Standard

Offline evaluations have the advantage that they do not require user interaction and are therefore often more easy to conduct. Instead the difficulty of offline evaluations lies in the dataset that is used for the evaluation. Usually such a data set consists of user-feedback that was manually generated by experts or collected from a live system. In the context of recommender system the dataset may contain user ratings for recommended items which present a gold standard. However, Wikipedia does not provide such gold standard dataset nor we have the resources to manually create it. But we can use other Wikipedia data to derive quasi-gold standards. Before introducing our quasi-gold standards, we briefly clarify terminology of gold standard and quasi-gold standard.

Terminology

The common approach of evaluating information retrieval systems is to compare the retrieved documents to a reference model to classify a document as either relevant or irrelevant. A gold standard or ground truth is the perfect reference model that provides the best possible responses to any tested query. True positives are all retrieved documents that are part of the gold standard and therefore relevant. All other retrieved documents are false positives, i.e. irrelevant. For many applications a gold standard remains a theoretical concept, which is impossible to achieve in real world. Even a traditional user study, in which domain experts are asked to identify relevant documents, cannot completely eliminate misjudgments especially false negative errors as experts may miss relevant documents. Therefore, we introduce the term quasi-gold standard as approximation of a perfect reference model. A quasi-gold standard provides relevance judgments of comparable quality as the relevance judgments of domain experts. Retrieved documents that are part of the quasi-gold standard are true positive. However, the quasi-gold standard cannot distinguish whether all other retrieved documents are false positive or false negative. In the context of Wikipedia, a quasi-gold standard is capable of determining if a retrieved document is a relevant recommendation for a topically related Wikipedia article, but not, if a recommendation is irrelevant.

Click Streams

In 2015 Wikimedia Research released a click stream dataset for the English Wikipedia [45]. The dataset consists of aggregated referrer information for Wikipedia articles for 11 month between January 2015 and January 2017. For instance, with the click stream data you can visualize the incoming and outgoing traffic to the "London" article (Figure 2.3).

More interesting for our use case, this data also allows to determine the number of clicks on out-links for the available articles. For out-links, which occur multiple times in an article, only the total number of clicks is provided. The number of clicks on a link can be considered as a judgment of relevance, because we assume that the more relevant a linked document is the more frequent its link gets clicked. Therefore, click stream data can also be used as quasi-gold standard for evaluating recommender systems. Instead of a binary relevance classification, which "See also" links enable, click streams allow a classification on a cardinal scale, i.e. the number of clicks per link.

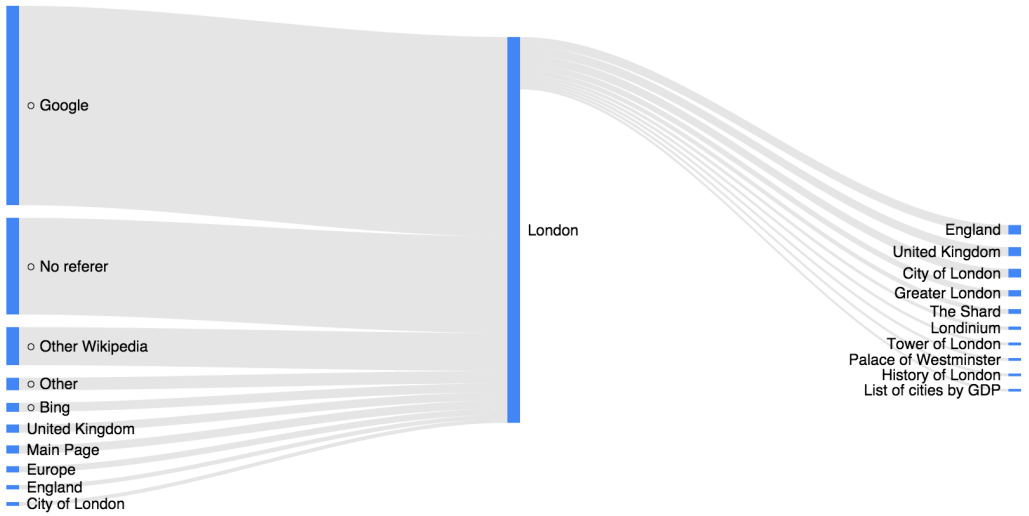

Figure 2.3: Visualization of click stream data. Incoming and outgoing traffic to the "London" article on English Wikipedia during January 2015.

Click streams are only provided in the English language. In order evaluate simple English and German Wikipedia with click streams, the dataset must be translated. In this done in pre-processing step as Flink job. The Flink job maps each English article title to its unique page id and then finds the article title in corresponding language from the Wikipedia inter-language link dataset[3].

"See also" Links

Aside from their main content, Wikipedia articles contain pointers to additional information in form of references or external links, but also a so-called "See also" section (Figure 2.4). The Wikipedia guideline states that this section should include a list of internal links to topically related Wikipedia articles. The purpose of "See also" links is to enable readers to explore tangentially related topics[4]. The links can assist readers in finding related articles. For our evaluation, we assume that "See also" links correlate with the expected results of a recommender system.

Referring to Manning [25], the "See also" links are a user-generated judgment of relevance, i.e. they are a quasi-gold standard. When using "See also" links as a quasi-gold standard, we can classify document relevance as follows:

[3] `https://en.wikipedia.org/wiki/Help:Interlanguage_links`
[4] `https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Layout`

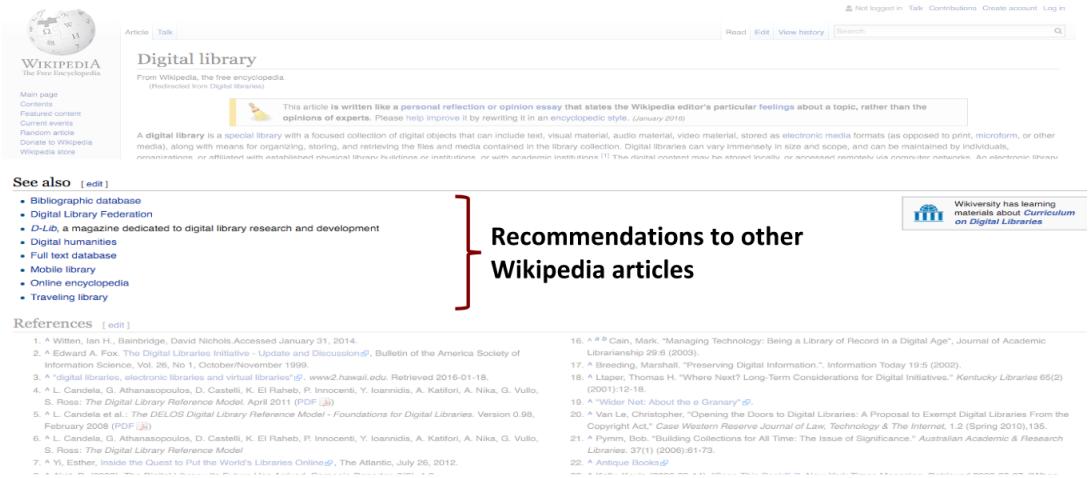

Figure 2.4: "See also"-links from the "Digital library" Wikipedia article.

The documents that the investigated similarity measures retrieved and that exist as "See also" links are judged as relevant. Retrieved documents for which no "See also" link exists are classified as irrelevant. At this point, we see a problem: We expect the "See also" links to be an incomplete gold standard, since Wikipedia's volunteers, whose main objectives might be creating textual content rather than providing literature recommendations, create this content. Even if a retrieved document cannot be found within the "See also" links, it still can be topically related, i.e. relevant. Therefore, we can decide if a result is relevant, but not if it is irrelevant. A true binary classification is not possible. Hence, we expect a precise true positive classification for documents that exist as "See also" links, while many results could be classified as false negative without document similarity measure failures, when the retrieved document are simply missed by "See also" links. Consequently, the performance measure should consider these properties of a quasi-gold standard by not excessively penalizing recommendations for documents that cannot be classified as relevant.

Similar to the translation of click stream data set from English to simple English and German, "See also" links are also combined across Wikipedia languages to increase the coverage of the gold standard.

Edit History

One of the many data dumps provided by Wikipedia contains information on what articles a user has edited, i.e. the edit history of Wikipedia authors[5]. We expected that this data can be also utilized as quasi-gold standard, if the following assumption is correct: Wikipedia authors tent to work on articles that are topically related.

Rank	Recommendation	Co-edits
Seed article	*List of planets*	
1	Fire	13
2	Evolution	12
3	Tom Kaulitz	12
Seed article	*Fast & Furious 7*	
1	Bernie Sanders	5
2	2014 North American polar vortex	4
3	2015 San Bernardino shooting	4
Seed article	*Dragonfly*	
1	Evolution	11
2	Business	8
3	Central processing unit	8

Table 2.1: Edit history recommendations (simple English). Co-edits is the number of authors who simultaneously work on the seed article and the recommended article.

To test this assumption, we generated recommendations from the edit history by extracting co-edited articles from authors of a given source article and then ranking the co-edited articles by the number of authors who work at the same on the source and co-edited article. A sample of the recommendations is shown in Table 2.1. From this and other samples we conclude that our assumption is wrong. The generated recommendations seem to have no topical relatedness with the seed article. For instance, the article of the member of a German pop band "Tom Kaulitz" is irrelevant as recommendation for "List of planets". Similarly, the politician "Bernie Sanders"

[5]Latest edit history dump from simple English Wikipedia (gz-File, +300MB): `https://dumps.wikimedia.org/simplewiki/latest/simplewiki-latest-stub-meta-history.xml.gz`

is irrelevant to the action movie "Fast & Furious 7". Therefore, we dispose the edit history as quasi-gold standard.

2.3 Co-Citation Proximity Analysis

The concept behind our recommender system is Co-Citation Proximity Analysis (CPA), which was introduced by Gipp and Beel in 2006 [1]. It was not originally developed for finding related Wikipedia articles, instead it originated from the field of library science. To be precise, original CPA aims to measure similarity of scientific documents based on co-citations proximity. While CPA itself is relatively new, because only in recent year more full-text documents got available, the idea of a co-citation-based similarity measure namely Co-Citation was already published in 1973 by Small and Marshakova-Shaikevic [42, 26].

Two documents A and B are in a co-citation relationship if cited by other documents C or D simultaneously (Figure 2.5). This has the advantage that the similarity measurement is not purely based on internal features as, for instance, in Bibliographic Coupling [27]. Instead, Co-Citation evaluates the citations that a document receives, i.e. external features, which are at the same time an indicator for relevance. The number of papers citing two documents together (co-citations) corresponds to the co-citation strength. The documents A and B in Figure 2.5 have the co-citation strength of two as both are co-cited by C and D.

CPA extends this concept by utilizing the additional information implied in the citation marker, i.e. the position of a citation within the text. CPA's idea is that, when citation markers of co-cited documents are in close proximity, the documents are more likely to be similar (Figure 2.6). The benefit of using co-citation proximity has been shown in several studies [24, 15, 21]. In their original publication, Gipp and Beel distinguished five levels of citation proximity based on the citation marker: same sentence, same paragraph, same chapter, same journal and same journal but different year. For instance, documents co-cited in the same sentence are more similar to each other than documents co-cited in the same paragraph. Depending on their proximity level, each co-citation is assigned with a fixed value called Co-Citation Proximity Index (CPI).

In our previous work [39], we derived a more general CPI model that can be also used for Wikipedia links (Equation 2.1). We defined a $m \times m$-matrix with element $\delta_{i,j}$ that stores the link position for all m documents. Specifically the column for document j, $\delta_{*,j}$ holds the positions for links to other documents in words counted

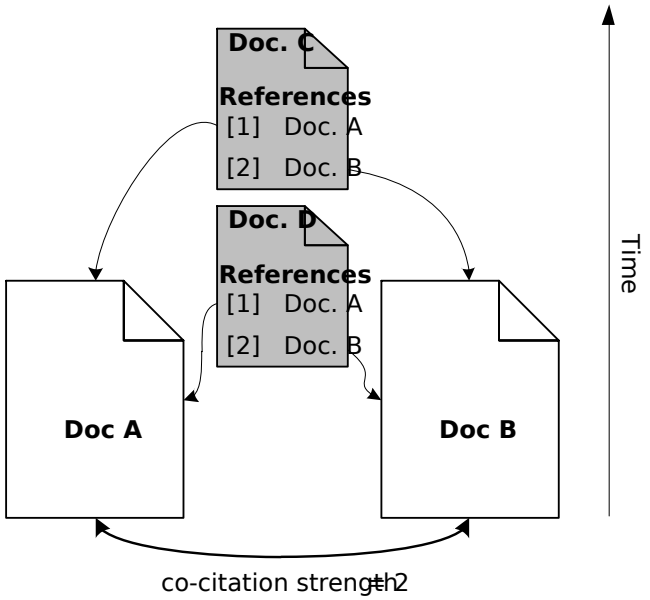

Figure 2.5: Co-Citation relationship between documents [17]. Documents A and B are both co-cited by C and D.

from the beginning oft the document. Then, the CPI for a document pair a and b is the sum over their link distance pairs in all m documents damped by a negative exponential parameter α, where α defines how the link distance is weighted.

$$\mathrm{CPI}(a,b) = \sum_{j=1}^{m} \Delta_j(a,b)^{-\alpha} = \sum_{j=1}^{m} \begin{cases} |\delta_{a,j} - \delta_{b,j}|^{-\alpha} & \delta_{a,j} > 0 \wedge \delta_{b,j} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

The two previously conducted offline evaluations [39] suggested the values $\alpha_1 = 0.81$ ("See also" evaluation) and $\alpha_2 = 0.9$ (click stream evaluation) as best parameter for the English Wikipedia.

2.4 Term Frequency - Inverse Document Frequency

Introduced by Jones and Salton [20, 37], Term Frequency Inverse Document Frequency (TF-IDF) is nowadays the theoretical basis for many IR applications. For instance, it is also the the conceptional foundation for MLT. TF-IDF is a term weighting schema, i.e. it defines the importance of a term in a document or a search

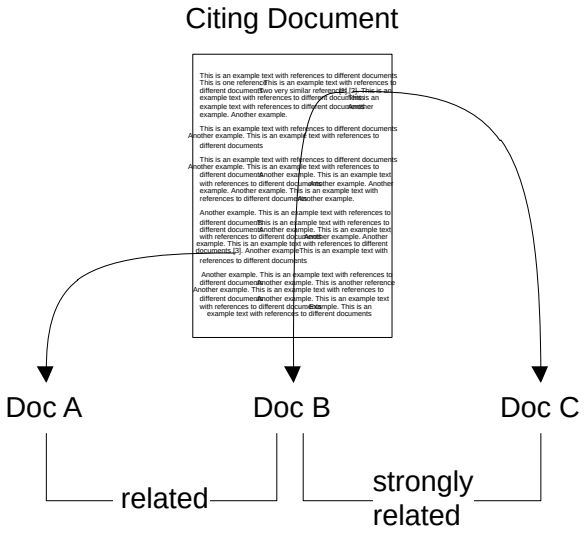

Figure 2.6: Co-Citation Proximity Analysis. Document B and C are stronger related than document B and A, since their citation markers are in close proximity [17].

query and therefore can be also used to measured the semantic similarity of documents. The term importance increases if the term frequency (TF) in a document in high while the term appears with low frequency in the corpus and vice versa. The appearance of a term in the corpus can be understood as its specificity, which can be quantified as an inverse function of the number of documents in which it occurs and is called inverse document frequency (IDF). In 1972 IDF was proposed by Jones [20] as heuristic method and later Robertson [34] provided a information theoretic justification for it.

Table 2.2: Common variants of IDF weighting schema

Weighting schema	**IDF weight**
Classic IDF	$log(\frac{N}{n_t})$
Smooth IDF	$log(\frac{1+N}{n_t})$
Probabilistic IDF	$log(\frac{N-n_t}{n_t})$
Okapi BM25 IDF [35]	$log(\frac{N-n_t+0.5}{n_t+0.5})$

An alternative approach to TF-IDF is Okapi BM25 [35]. Okapi BM25 is based on a probabilistic model for document retrieval. If it is known that a term i occurs

in n documents within a collection of size N, then the probability of any random document containing the term i is $\frac{n}{N}$. The log of this probability gives the information content of the word. This probability model has a strong theoretical basis for finding the most likely relevant documents and provides also an IDF term.

Table 2.2 shows four variants including Okapi BM25 of inverse document frequency functions for the term t, where N is the total number of documents in the corpus D ($N = |D|$) and n_t is the total number of documents that contain term t. These variants are later used to derive the Inverse Link Frequency factor for CPA.

2.5 Elasticsearch

Elasticsearch, developed by Elastic an US-based company, is a full-text search engine build on top of Apache Lucene [28]. It is written in Java and available under Apache's open source license. Elasticsearch comes with all of Lucene's search functionalities including the MoreLikeThis component, which can be utilized as text-based recommender system. Because of its distributive and scalable design, Elasticsearch is also used by the Wikimedia Foundation to provide a search feature for Wikipedia (Section 2.6). For ranking the search results Elasticsearch relies on the concept of TF-IDF and the Vector Space Model [36]. Aside Elasticsearch search functionalities, we use it as key-value store to retrieve CPA recommendations that are pre-computed with Apache Flink (Section 4.3).

2.5.1 Scoring Function

In the following we introduce the scoring function that is used by Elasticsearch (Lucene) to rank search results according to their relevance to the search query. The scoring function is as well used for the MoreLikeThis feature. A detailed description can be found in Elasticsearch documentation [6].

$$\begin{aligned} \text{score}(q,d) = \text{queryNorm}(q) \times \text{coord}(q,d) \sum_{t \in q} \text{tf}(t,d) \times \text{idf}(t)^2 \\ \times \text{boost}(t) \times \text{norm}(t,d) \end{aligned} \tag{2.2}$$

[6] https://www.elastic.co/guide/en/elasticsearch/guide/current/practical-scoring-function.html

The score function score(q, d) represents the relevance score of a document d for the query q (Equation 2.2), whereby candidate documents are retrieved in a first step with Boolean query. The factors of the score function are defined as:

1. $tf(t \in d) = \sqrt{\text{frequency}}$ is the term frequency for term t in document d.
2. idf$(t) = 1 + log(\frac{|D|}{f_t+1})$ is the inverse document frequency for term t, whereby f_t is the document frequency and $|D|$ is the total number of documents in the corpus.
3. queryNorm(q) is the query normalization factor.
4. coord(q, d) is the coordination factor.
5. boost(t) is the boost that has been applied to the query.
6. norm(t, d) is the field-length norm, combined with the index-time field-level boost, if any.

For the CPA recommendations we also rely on the score function, whereby the TF-IDF and query factors at set to one and only the boost factor is used to rank documents according to their recommendations score (Section 4.3).

2.5.2 MoreLikeThis

Elasticsearch's MoreLikeThis (MLT) feature works similar to its keyword search engine[7]. MLT relies on the same inverted index and scoring function (Equation 2.2). However, in contrast to the keyword search a seed document, for which similarity documents should be returned, is used as query input. For each term of the query document the TF-IDF weight is computed. Then, all terms are then sorted accordingly and the top terms (by default 25) are selected and used to query the search index, whereby the boost factor is set to the TF-IDF of the corresponding term. Finally, all retrieved documents are then ranked based on the score function.

$$\text{score}_{\text{Wikipedia}}(q, d) = \text{score}(q, d) \times \text{popularity}(d) \tag{2.3}$$

Wikipedia modified MLT by including an additional popularity factor, which is based on page views on document d, in the final score function (Equation 2.3). By

[7] `http://cephas.net/blog/2008/03/30/how-morelikethis-works-in-lucene/`

doing so, Wikipedia solves a problem that we found in our previous study [39]. That is that MLT tends to retrieve niche articles that are barely relevant. The popularity factor penalizes those niche articles.

2.6 MediaWiki

MediaWiki is an open source Web application for Wikis written in PHP, originally for use on Wikipedia. It is now also used by several other Wikimedia Foundation projects and third-parties.

MediaWiki-Vagrant MediaWiki-Vagrant is a portable MediaWiki development environment. It allows to simulate the Wikipedia productive environment in a virtual machine. We use MediaWiki-Vagrant to pre-test all code locally before deploying it to a server.

CirrusSearch CirrusSearch is a MediaWiki extension that uses Elasticsearch to provide enhanced search features over the default MediaWiki search. Besides keyword based searching, CirrusSearch offers also a series of prefix queries with additional filtering functions. For instance, using the prefix *incategory:Music* would return only articles from the "Music" category as search results. MoreLikeThis is integrated in the same fashion. With *morelike:Water* all text-based recommendations for the "Water" article are returned. The Wikimedia Foundation uses CirrusSearch for all its Wikimedia projects. For the integration of our CPA recommendation, we also rely on a modification of the CirrusSearch extension (Section 4.3).

To sum it up, MediaWiki and its extensions are the center-piece of the Wikipedia website and, in order make the CPA recommendations accessible to Wikipedia's users, we must work with the MediaWiki source code.

2.7 Apache Hadoop & Flink

With its several million articles and large amounts of logging data, Wikipedia's data exceeds the capabilities of a single computer. Thus, techniques for distributed computing and storage are needed.

Apache Hadoop Hadoop[8] is a common open source framework for this task. It consists mainly of two components: Hadoop Distributed File System (HDFS) for fault tolerant distributed storage with a master-slave architecture and Hadoop MapReduce, an implementation of the MapRuce programming model for "Big data" processing in parallel.

Apache Flink Formerly known as Stratosphere and started as research project of TU Berlin, Apache Flink[9] is a part of the Apache Software Foundation [11, 2]. Apache Flink is, like Hadoop, an open source Java framework for processing "Big Data". It is capable of batch and streaming data processing. Yet, Flink's architecture is based on streaming model, it iterates data by using streaming. Its pipelined architecture allows processing the streaming data faster with lower latency than micro-batch architectures as in other frameworks like Apache Spark [40]. Flink does not come with its own storage engine, but it can be built upon a distributed file systems like HDFS. Figure 2.7 illustrates the layer architecture of Apache Flink[10]. Flink is written in Java and Scala, whereby we use Flink's Java API[11] from version 1.1.

Hadoop's MapReduce programming model enables processing of large datasets and is suitable for many real world use cases. Nevertheless, writing efficient application in MapReduce requires strong programming skills and in-depth knowledge of its architecture. Apache Flink has been developed, in order to allow non-experts to use such systems, save development time and make application code easier to understand and maintain.

The goal of this thesis is the deployment of a Flink-based recommender system into the Wikipedia infrastructure. Thus, an optimized implementation is essential. Therefore, Apache Flink offers us the opportunity to benefit from its "Big Data" technologies such that we can focus on high level performance optimizations. Other frameworks like Apache Spark offer similar optimizers and, therefore, they would be also suitable for the development of a large-scale recommender system.

[8] `https://hadoop.apache.org/`

[9] `http://flink.apache.org/`

[10] `https://ci.apache.org/projects/flink/flink-docs-release-1.1/internals/components.html`

[11] `https://ci.apache.org/projects/flink/flink-docs-release-1.1/`

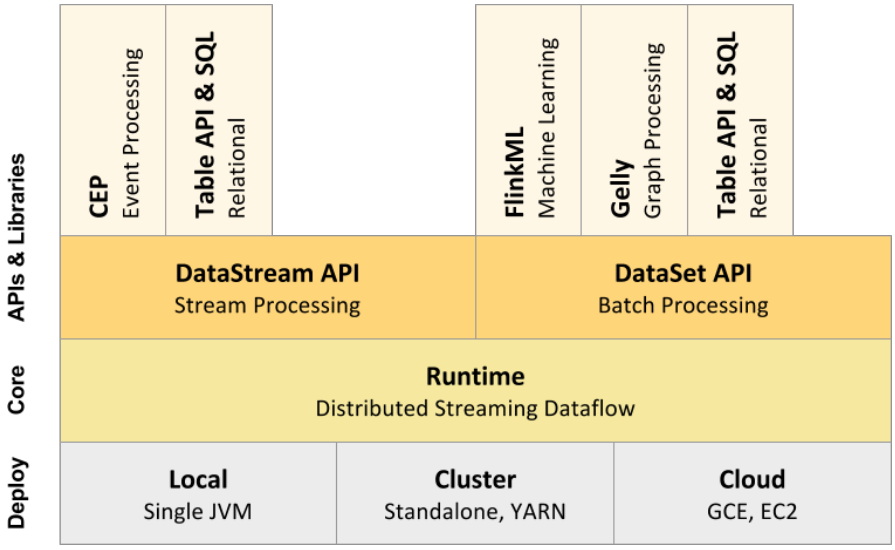

Figure 2.7: Apache Flink layer overview. The DataSet API (Batch processing) is suitable for processing large data chunks like Wikipedia dumps.

2.8 Apache Oozie

Wikipedia's authors frequently create new or rewrite old articles. Consequently, the generations of recommendations needs to take place in frequent time intervals to reflect content changes. For this reason, a scheduler is needed to manage the recommendation generation. Apache Oozie [19] is a Web application written in Java for scheduling Apache Hadoop jobs. Oozies allows the administration of multiple jobs as a single logical unit. It is integrated with the Hadoop stack, with YARN as its architectural center, and supports natively Apache MapReduce or Apache Spark jobs. Due to Flink's abilities of running in a YARN environment, Flink jobs can be executed as well with Oozie. Moreover, Oozie can also schedule system-specific jobs, like Java programs or shell scripts.

The Oozie web applications provides greater control over jobs and also makes it easier to repeat those jobs at predetermined intervals. Aside job scheduling at fixed time frame, jobs can be scheduled based on the existence of data in HDFS (e.g. newly available Wikipedia data). Hence, Oozie is suitable for the task of frequent recommendation generation when a new Wikipedia dump gets available.

There are two basic types of Oozie jobs:

1. Oozie Workflow jobs are Directed Acyclical Graphs (DAGs), specifying a se-

quence of actions to execute. The Workflow job has to wait

2. Oozie Coordinator jobs are recurrent Oozie Workflow jobs that are triggered by time and data availability.

The Wikimedia Foundations relies on Apache Oozie for all data processing tasks within their Wiki projects. This involves jobs like the aggregation of page view data. But also a job exist to read from HDFS and populate the data into Elasticsearch. When we want to make the CPA recommendations available to CirrusSearch, this is exactly what we need to do. Therefore, we can reuse the code of this job, write a new job and integrate it into Oozie such that it can be run frequently.

2.9 Performance Measures

In our offline evaluation the performance of CPA is measured with a combined objective function for the top-k recommendations. The top-k recommendations with $k = 3$ are evaluated, since in the online experiment, i.e. the Wikipedia Android app, are also only three recommendations presented to the user. The performance measures used by the objective function are: Mean Average Precision for the "See also"-link evaluation, Click Trough Rate for click stream evaluation and two recommendation coverage metrics for both quasi-gold standards.

Mean Average Precision

The rank-based performance measure Mean Average Precision (MAP) is widely used among the information retrieval community.

$$\text{MAP} = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} \text{Precision}(R_j k) \tag{2.4}$$

For a single query or seed document, Average Precision is the average of the precision value obtained for the set of top k documents existing after each relevant document is retrieved, and this value is then averaged over all queries. That is, if the set of relevant documents for a query $q_j \in Q$ is $\{d_1, \dots d_{m_j}\}$ and R_{jk} is the set of ranked retrieval results from the top result until you get to document d_k, then MAP is defined as in Equation 2.4.

We use for the "See also" link evaluation on MAP as performance measure, whereby all articles with "See also" links correspond to the queries Q and retrieved documents are classified as relevant if they are part of the "See also" links.

Click Through Rate

The Click Through Rate (CTR) is the ratio of users who click on a specific link to the number of impressions, i.e. the number of total users who view the link (Equation 2.5). It is commonly used to measure the performance of online advertisements. But also in the context of RS the CTR can be understood as relevance judgment: The more often a recommendation is getting clicked, to more relevant it is. But CTR should be used with precaution. For instance, Zheng et al. [46] showed that CTR and relevance do not always correlate and concluded that "CTR may not be the optimal metric for online evaluation of recommender systems".

$$\text{CTR} = \frac{\text{Clicks}}{\text{Impressions}} \times 100 \tag{2.5}$$

In our offline experiment with the click stream data we use the CTR as performance measure. However, we normalize the metric such that in the optimum CTR is equal to 1 (Equation 2.6). Therefore, we do not use the ratio of impressions, instead we divide the number clicks on our top-k recommendations by the number of clicks that the top-k of the most clicked out-links received, i.e. the theoretical optimal recommendations.

$$\text{CTR}_{\text{opt}} = \frac{\text{Clicks on Recommendations}}{\text{Clicks on Most Clicked Out-Links}} \times 100 \tag{2.6}$$

Recommendation Coverage

The use of recommendation coverage and serendipity as performance measures gets more attention from the RS community. The on-going debate on filter bubbles and echo chambers will also enhance this trend. While coverage can be understood as the degree to which the recommended items cover the set of all available items, serendipity is concerned with the novelty of recommendations and how they may positively surprise the user [16].

With this regard, we also make use of two metrics that represent the recommendations coverage. First, there is the inverse number of recommendations of the most

recommended article. For instance, let the "United States" article be with 1000 recommendations the most recommended article, then the corresponding score would be $\frac{1}{1000}$. Second, we measure the distinct recommendation $\frac{|R_{distinct}|}{|D|}$ that is the number of distinct recommendation $R_{distinct}$ in ratio to the total number of available articles D. Both recommendation coverage scores are normalized, i.e. in the optimum each score is equal to 1. Moreover, the scores can be computed independently from our quasi-gold standards.

Combined Objective Function

Quantifying the performance of recommender system is generally difficult due to the subjective nature of recommendation. Yet, we need to decide for specific parameters of the CPI model. Thus, we define an objective function $f(\theta, D)$ that combines the four different performance measure as in Equation 2.7, where D is the document corpus and θ are the model parameters.

$$\begin{aligned} f(\theta, D) = 0.45 \times \mathrm{CTR}_{\mathrm{opt}}(\theta, D) + 0.45 \times \mathrm{MAP}(\theta, D) \\ +0.05 \times \frac{|R_{\mathrm{distinct}}|}{|D|} + 0.05 \times \frac{1}{|R_{\mathrm{top}}|} \end{aligned} \tag{2.7}$$

Each measures is weighted according to our subjective judgment. We account the "See also" and click stream evaluation as most important and therefore we weight both with 45% each, whereas the two recommendation coverage measures are weight with 5% each because we consider them as less important.

2.9.1 Online Metrics

For the evaluation we rely on performance measures that can be derived from Wikipedia's event logging system. Hence, CTR (Equation 2.5) is the primary performance measure, whereby a click is defined as the process of a user clicking on a recommended article and a view as a user viewing a set of recommendation. However, we are aware that a click on a recommendation does not necessary mean that the recommended article is relevant. A user might click on an article because of an exciting title but as soon as the user reads the first sentence he or she goes back to the previous article. For this reasons we also measure *Long Clicks*. As long clicks we define clicks that make the user spending at least 10 seconds on the recommended article and viewing at least 50% of the article.

In addition, we also measure the time spent (in seconds) on an article and the maximum viewed of an article (in percentage). The latter two performance measures are evaluated because for Wikipedia the overall objective on a new feature is to increase the user-engagement. An increase in clicks on recommendations does not necessary lead to an overall increase in user-engagement. Instead, clicks on other features might decrease.

Discussion In this chapter we discussed the concepts of the text- and link-based recommender systems, relevant technologies like Elasticsearch and Apache Flink, the Wikipedia data set and performance measures that we use for this work. All these background informations represent the foundation for development and evaluation of the investigated link-based recommender system. Before we describe in detail the research approach, we first give an overview about related work that is relevant to our research.

Chapter 3

Related Work

In this section we present related work from recommender system research whereby we focus on evaluation studies and papers that address evaluation-related issues. We refer to investigation of the effect of co-citation proximity. We start this section with a summary of the preliminary study that is the foundation of this thesis.

3.1 Preliminary Study

Schwarzer et al. [39] were the first to apply the idea of CPA on Wikipedia. They transfered the concept from citations in scientific papers to links in Wikipedia articles. Thereby, they introduced a general approach to compute proximity (Equation 2.1). Schwarzer et al.'s offline evaluation was based on Wikipedia's "See also" links and click streams. Recommendations from MLT, CPA and classical co-citation analysis were compared for English Wikipedia articles. But from the results of the study, they could not draw a clear conclusion regarding the superiority of one recommendation algorithm. MLT and CPA seemed to perform similarly well.

As shown in Figure 3.1, CPA achieves the best result with absolute clicks, whereas MLT outperforms CPA in terms of CTR. Yet, the positive effect of proximity on co-citation analysis is visible through out all evaluations, since CPA always outperforms classical co-citation analysis. Due to the promising results of CPA, they state that link-based recommendations need to be subject to future research.

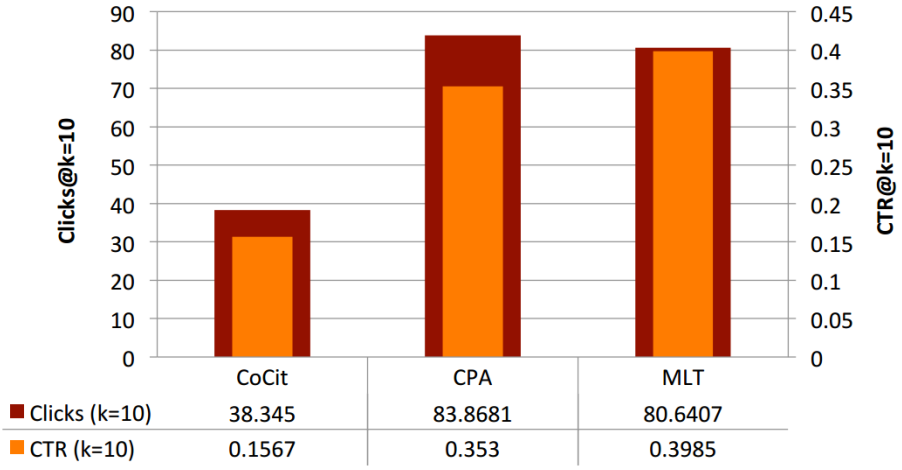

Figure 3.1: Results of preliminary study. Click stream evaluation with absolute clicks and Click-Trough-Rate (CTR) based on 2,5M articles.

3.2 Recommender System Evaluations

Gunawardana & Shani review in their paper from 2009 [18] the proper construction of offline experiments and the corresponding accuracy metrics for deciding on the best performing recommender system. They discuss evaluation metrics like Precision-Recall or Root of the Mean Square Error (RMSE) and general recommendation techniques like predicting ratings. Gunawardana & Shani explicitly emphasize the importance of user interface design and the role of the recommendation in an application. Researchers need to take into account whether a recommendation system is only a supporting system or the main component. Furthermore, they empirically demonstrate with different data sets that in some cases two algorithms can be ranked differently by two metrics over the same data set. Thus, we should not focus on a single metric.

In 2016 Beel et al. [5] investigated the reproducibility of recommendation system research. Even if reproducibility should be a key element of scientific research, Beel et al. found that recommendation system research often lacks reproducibility. Minor variations in approaches and scenarios, e.g. a different test collection, can lead to significant changes in a recommendation performance and therefore reproducibility is difficult to ensure. For future research they propose for instance to foster the development and use of recommendation frameworks and to establish best-practice

guidelines for recommendation system research. From Beel et al.'s findings we derive for this work that our recommender system should be evaluated with at least two test collections.

Moreover, Beel & Langer [7] published one of the few studies from the field of research paper recommender systems which makes use of three evaluations methods, i.e. offline evaluation, online evaluation and user study. They investigate in detail how the evaluation methods effect the outcome and show that the results also can be contradictory. For the evaluation Beel & Langer rely on data from the literature management tool *Docear* [8]. Regarding the evidence of offline evaluation, they doubt that "researchers will ever be able to reliably predict whether human factors affect the predictive power of offline evaluations" and recommendation that ideally a recommendation system with evaluate with both online evaluation and user-study. In the case of their online evaluation they furthermore conclude that Click-Trough-Rate is the most appropriated evaluation metric for their application. However, this is not generally true as they state. Other use cases may favor other evaluation metrics.

Dooms et al. test in [13] an event recommender system on a popular Belgian cultural website. They apply content-based, collaborative filtering and hybrid approaches, whereby the hybrid recommendation system generates the most promising results. As evaluation Dooms et al. conduct both user-study and offline evaluation even if they state "offline evaluation metrics are inadequate for this task". In the user study the participants were ask to give explicit feedback in an questionnaire of 14 questions regarding the recommendation performance. The comparison of the offline and user study results reveals a small change in the ranking of the recommender system. Consequently, we should not neglect the validity of our evaluation even if the contradict each other.

Bambini et al. [3] compare collaborative and content-based recommendation system for television programs and video-on-demand content in the IPTV service, *FastWeb*[1]. They evaluate both recommendation techniques with an accuracy-oriented offline evaluation and online evaluation. In the offline evaluation Bambini et al. also evaluate serendipity by doing a long-tail test, i.e. they excluded the top-10 items from the recommendation lists and then measure the recall. In the online experiment the user feedback is measured in terms of the percentage of views that have been triggered by the recommendation system. In addition to the direct effect of the recommendation system, they also measure the overall impact of the usage of the IPTV service, i.e. lift factor regarding the total number of views and sales.

[1] `http://www.fastweb.it/`

In 2007 Ollivier & Senellart [31] compared Green Measure to several other methods for finding related pages in the case of the English version of Wikipedia. They found out that Green Measure has both the best average results and the best robustness compared to Co-Citation, Cosine similarity with TF-IDF, PageRank of Links and Local PageRank. A user study measured the performance of each method.

In the study from 2010 Davidson et al. [12] demonstrate how they developed and tested a video recommender system for *YouTube*. Video recommendations are generated by using a user's personal activity as seeds and expanding the set of videos by traversing a co-visitation based graph of videos. They rank the recommendations using a variety of signals for relevance and diversity. The evaluation is performed in online manner whereby the data collected using A/B tests from within the *YouTube* production system. *YouTube* users were diverted into distinct groups where one group acts as the control or baseline and the other group is exposed to the novel recommender system. The two groups are then compared against one another over click rates.

3.3 Effect of Co-Citation Proximity

The effect of co-citation proximity is gaining more attention from the research community in the recent years. This is mainly due to the increased availability of full-text documents.

For instance, Knoth et al. [21] tested the use of CPA as practical recommender systems for research papers. They developed a scalable recommender systems from a corpus of 368,385 full-texts articles. In their experiments Knoth et al. quantify the proximity as minimum, sum or mean of characters between co-citations smoothed with logarithm and normalized with the co-citation count. With an user survey they show that CPA can outperform classical co-citation recommendations when using sum or mean as proximity function.

Beel et al. [4] tested in a study on research paper recommender systems if the effect of co-citation proximity can be further improved. In particular, they tested if the CC-IDF citation-weighting schema, an adaptation from Inverse Document Frequency, is beneficial. Their click-oriented evaluation revealed that CC-IDF was not more effective than classical citation-weighting.

With their work on a large-scale recommendation system for the biomedical knowledge base *meta.com*, Perrie et al. [32] proved that a CPA-based recommender system can be used with corpus of 27 million articles in a production environment.

However, they stress that one problem is that CPA fails to generate recommendations for 25% of the papers in their experiment.

Discussion The review of related worked shows that recommender system are a popular subject among the information retrieval community. Past studies address common issues of recommender system evaluation. We want to learn from these studies. For example, we can use two test collections and test different performance metrics to check the variation of the recommender algorithms. Moreover, we see that offline evaluation are often conducted even if a correlation to real user feedback is not naturally given. Hence, we aim to conduct an online evaluation for which A/B testing is a common procedure. With respect to CPA, the preliminary study and as well as the literature provides promising results which require future investigations that we want to contribute with this thesis. At the same time, we aim to resolve CPA issues like the missing ability of generating recommendations for articles.

Chapter 4

Approach

In this chapter, we present our approach, explain the developed system architecture and describe the required steps in detail. Since this is a follow-up research, we focus on elements that we implemented as part of this thesis. The implementation of the previous study [39] consisted mainly of a Flink job, which was capable of reading a Wikipedia XML dump, generated CPA recommendations with resolved redirects and writing the recommendation sets to HDFS as CSV file, and additional Flink jobs for the offline evaluation. For this thesis, the whole source code was reviewed, optimized and integrated into the Wikipedia system in order to perform a user-centric instead of a pure offline evaluation. While cooperating with Wikimedia Foundation, our work was presented as "Citolytics" project to the Wikipedia community [38].

4.1 System Overview

To provide orientation, we first give an overview of the developed system and its components. The system is presented in Figure 4.1. Essential for our evaluation approach is the modification and access to four components from the Wikipedia system: (1) The Android application needs to be modified to display the link-based recommendations. (2) We must read from the EventLogging system, where the tracked user behavior from the Android app is stored. (3) The CPA recommendations needs be made accessible via the MediaWiki API, which can be done by modifying the CirrusSearch extension. (4) The Flink job needs to be integrate into the Oozie pipeline and its results need to be written to Elasticsearch.

Our work regarding each component is explained in the following sections. Sec-

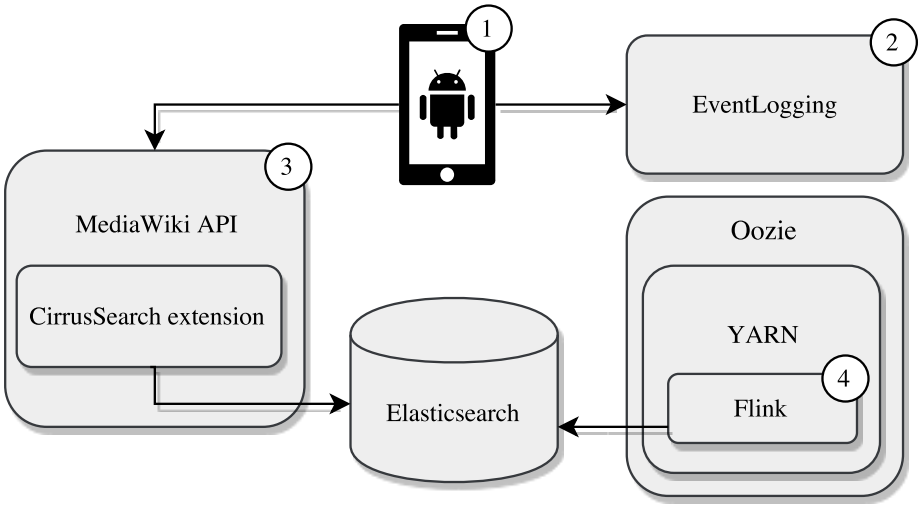

Figure 4.1: Overview of Wikipedia's system architecture and the components that are needed to deploy the recommender system.

tion 4.2 describes the optimization of the Apache Flink job. Our modification of the CirrusSearch extension is disclosed in Section 4.3. The EventLogging system is introduced in Section 4.5. Section 4.7 presents the integrations of recommendation into the Android app.

4.2 Apache Flink Job for Link-based Recommendation Generation

In order to make the recommender system meet the requirements of a production system, additional feature were added and the runtime performance was optimized. Figure 4.2 illustrates the processing steps. The major changes are explained in the following paragraphs.

Stop Links Removal

By analyzing the results of the preliminary offline evaluation, we found that links, which are not part of the main article content, can negatively affect the recommendations. Thus, an additional pre-processing step removes such links, for instance, from info boxes. This step can be considered as similar to the stop words removal

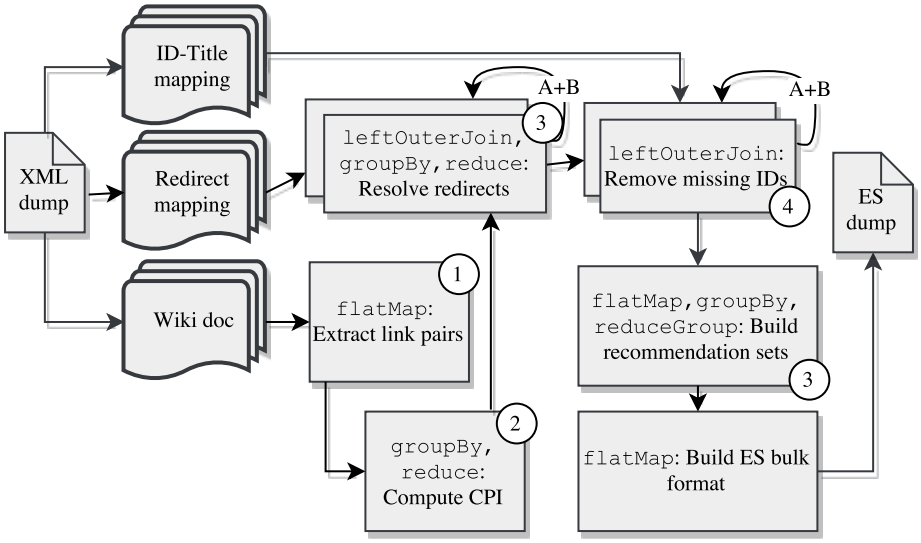

Figure 4.2: Schema of the Apache Flink job that generates the link-based recommendations.

step in classical text-based recommender systems. In Figure 4.2 stop links removals is part of the extraction of link pairs (step 1).

ID Validation

The original implementation was purely based on the links that were extracted from the XML dump. But Wikipedia allows authors to create links to article that do not exist and therefore the output also contained recommendations for non-existing articles, which are obviously irrelevant for our use case. Thus, we added a post-processing step that checks whether each recommendation exist as article, i.e. has a valid Wikipedia ID (Figure 4.2, step 4).

CirrusSearch Output

The Wikipedia setup requires the export of the recommendations to Elasticsearch from where CirrusSearch can access them. In agreement with Wikipedia developers, we modified the Flink job such that the output can be written in Elasticsearch JSON bulk format[1]. The bulk format requires article IDs to be included. The ID

[1] `https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html`

validation step provides these IDs. The usage of the Flink Elasticsearch data sink was not possible, since the Wikipedia systems for generating the recommendations and CirrusSearch are separated.

Testing

All the additional features are making the code more complex and thus harder to maintain and to hunt bugs. For this reason, we put special emphasis on unit testing. However, we found that there is no official unit testing framework for Flink. Also, the third-party implementation by Otto Group[2] was not suitable. Furthermore, we added the continuous integration tool TravisCI[3] to the project, whereby we found that memory expensive Flink unit test could not be handled by TravisCI.

Backup Recommendation

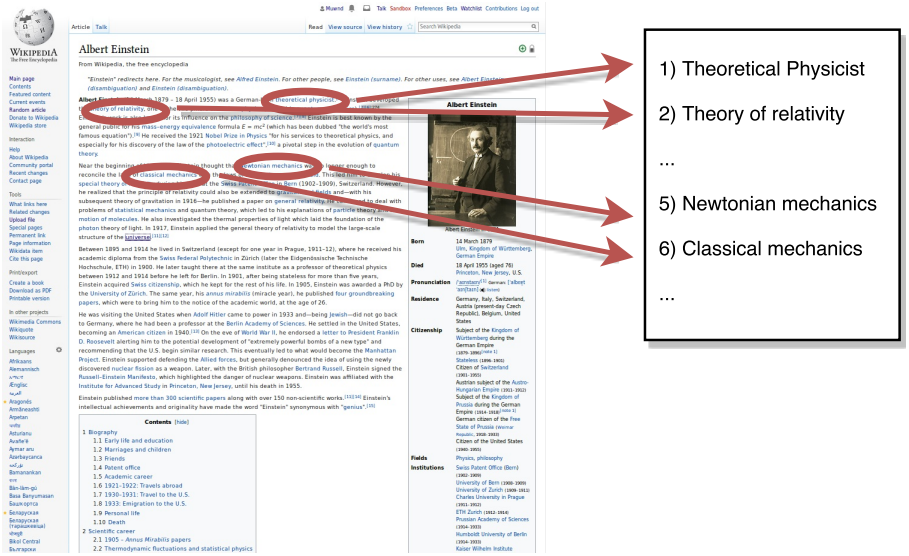

Figure 4.3: Backup recommendations are generated from out-links and ranked according to the link position.

As Perrie et al. [32] already revealed, one drawback of the link-based method CPA is that it cannot generate recommendations which have not any in-links from other

[2] https://github.com/ottogroup/flink-spector
[3] https://travis-ci.org/wikimedia/citolytics

articles. Depending on the Wikipedia language the amount varies (see in-links plots), e.g. approximate 10% articles in the simple English do not have any in-links. For a recommender system such blind spots are a disadvantage. To limit the number of missing recommendations, we implemented an additional backup strategy for those articles: We extract out-links from the article that has not enough CPA recommendations and rank them according to the link position, since we assume that links that are closer to the beginning of the article are more likely to be relevant. For instance, "Theoretical Physicist" is the top backup recommendations are the article "Albert Einstein" because is the the first link in the article (Figure 4.3). Independent from the position, backup recommendations are always ranked below CPA recommendations. When using backup recommendation the number of articles, for which recommendations are available, increases for German from 1,815,508 to 2,003,264 (+10%) and for simple English from 106,090 to 121,653 (+14%). Backup recommendations are part of the recommendation set building step (Figure 4.2, step 3).

Discussion All the mentioned improvements make the Flink job ready to be deployed in a production environment. The corresponding code of the Flink job can be found on the Citolytics GitHub repository[4].

4.3 CirrusSearch Integration

With the modification of the CirrusSearch extension we want to achieve that the Citolytics recommendations become accessible via the Wikipedia API. The Elasticsearch index of CirrusSearch stores all search-relevant article information, e.g. title, content and others (Listing 1).

CirrusSearch can perform classical keyword queries but also filter queries on this index. The CirrusSearch's abstraction of these query types is called *KeywordFeature*. A *KeywordFeature* allows the modification of the Elasticsearch query if a certain keyword is part of the query string. For instance, if *incategory:* is used as prefix in a query string, a boolean filter for the category field is added to the query. In the same fashion, we implement the *CitolyticsKeywordFeature*.

In order to access the recommendation sets, we add an extra *citolytics* field to each article in the Elasticsearch index (Listing 2). The *citolytics* field holds all information necessary for generating the corresponding query, which is the title of the

[4] `https://github.com/wikimedia/citolytics`

```
// /wiki_content/page/1
{
   "title": "Albert Einstein"
   "content": "...",
   "category": "...", // ...
}
// /wiki_content/page/2
{
   "title": "Newspaper"
   "content": "...",
   "category": "...", // ...
}
// ...
```

Listing 1: ElasticSearch index for Wikipedia articles in CirrusSearch.

```
// /wiki_content/page/1
{
   "title": "Albert Einstein"
   "content": "...",
   "category": "...",
   "citolytics:" [
     { "title": "Physics", "score": 1.5 },
       { "title": "Germany", "score": 1.2 }
   ], // ...
}
// ...
```

Listing 2: ElasticSearch index for Wikipedia articles with Citolytics recommendations.

recommended article and the recommendation score (CPI). Other options for storing the recommendations like a separated index were discussed but rejected by Wikimedia Foundation developers. The query prefix *citolytics:* triggers the *CitolyticsKeywordFeature*. When, for instance, the query string is *citolytics:"Albert Einstein"*, the recommendations for the article "Albert Einstein" should be returned. This is accomplished with two steps: First, the article for the recommendations are requested is queried with an exact match query on the *title* field. Second, based on the information provided in the *citolytics* field the main query in generated that filters by the recommended titles and then uses boosting[5] for sorting the recommendations

[5] https://www.elastic.co/guide/en/elasticsearch/guide/current/query-time-boosting.html

by their scores. The resulting query is shown in Listing 3. The corresponding pull request for the *CitolytisKeywordFeature* can be found on Wikipedia's code review platform Gerrit[6].

```
{
  "query": {
   "bool": {
     "should": [
       {
         "term": {
           "title.keyword": { "boost": 1, "value": "Physics" }
         }
       },
       {
         "term": {
           "title.keyword": { "boost": 0.5, "value": "Germany" }
         }
       }
     ],
     "minimum_number_should_match": 1,
     "filter": [ ]
   }
 }
}
```

Listing 3: Simplyfied ElasticSearch query for retrieving recommendation for the "Albert Einstein" article.

4.4 Apache Ooozie Integration

The integration of the link-based recommendations into Wikipedia's Oozie pipeline allows the frequent generation of new recommendation as soon as Wikipedia's content is updated. For this purpose, we established the following procedure in agreement with the Wikipedia developers:

A cron job frequently checks the Wikipedia XML dump website[7] for new available dumps. When a new dump gets available, it is downloaded and written to HDFS. A Ooozie Coordinator waits for the dump to be written to HDFS and then starts the Oozie Workflow. The Oozie Workflow consists only of two actions: First, it executes a shell script, which launches the Flink job. Then, a PySpark script reads the output

[6] https://gerrit.wikimedia.org/r/#/c/329626/
[7] https://dumps.wikimedia.org/enwiki/

from the Flink job, which is in Elasticsearch bulk format and located on HDFS, and sends the output to Elasticsearch.

4.5 EventLogging

To test the performance of the recommender systems, we need to evaluate the data that is collected in the A/B test in the Android app. Wikipedia's EventLogging system[8] (EL) is a tool for collecting this data. EL is used across several Wikimedia projects for modeling, logging and processing analytics data. Also, the Android app supports EL, i.e. at certain events the app send JSON-decoded log data via a HTTP request to Wikipedia's EL server. For modeling log data fixed schema are defined[9] of which three are relevant for our evaluation. The EL schema are presented in the Appendix.

4.6 CPI Optimization

Aside technical improvements, this thesis has also the objective to find out whether the concept of CPA, i.e. the CPI model, can be further improved. For this reason we describe in the following different approaches to optimize the CPI model.

4.6.1 Proximity Definitions

When computing recommendations based on the proximity of co-links, the quantification of the proximity is essential. In the previous study [39] we quantified the co-link proximity as distance in number of words between the two links (Equation 2.1), which we call in the following the absolute proximity $\Delta_j^{\text{absolute}}(a, b)$. However, we see two alternative definitions:

Relative Proximity Wikipedia consists of articles with variable length (Figure 2.1). Co-links in a long article can be further apart and therefore result in a lower proximity score then co-links in a short article of only a few paragraphs.

$$\Delta_j^{\text{relative}}(a, b) = \frac{|\delta_{a,j} - \delta_{b,j}|}{|j|} \tag{4.1}$$

[8] `https://wikitech.wikimedia.org/wiki/Analytics/Systems/EventLogging`
[9] `https://meta.wikimedia.org/wiki/Research:Schemas`

To capture this, we define the relative proximity $\Delta_j^{\text{relative}}(a, b)$ in a document j for the co-link of a and b that is their absolute proximity $|\delta_{a,j} - \delta_{b,j}|$ normalized with the total number of words of the linking article $|j|$ (Equation 4.1).

Structure Proximity Gipp and Beel proposed in the original publication of CPA [1] the usage of predefined proximity levels to find related scientific papers, e.g. for a co-citation from within a sentence account $\frac{1}{2}$, for paragraph-level $\frac{1}{4}$ etc. up to journal-level co-citations. We argued in [39] that this can not be transfered to Wikipedia due to the non-uniform structure of Wikipedia articles. Anyhow, this hypothesis needs verification and therefore we perform our evaluation also with a proximity $\Delta_j^{\text{structure}}(a, b)$ based on the article structure that is defined as in Table 4.1. The scores differ from the original proposal to capture the different proximity levels of Wikipedia articles compared to scientific papers.

Table 4.1: Definition of structure-level proximity that is used for CPI computation.

Co-link Level	**Score**
Paragraph	1/4
Subsection	1/8
Section	1/12
Article	1/20

To find out which proximity definition is superior, we compare absolute, relative and structure proximity in an offline evaluation.

4.6.2 α-parameter

In addition to the proximity definition, finding the best α-parameter is also part of the CPI optimization. The α-parameter defines in our CPI model (Equation 2.1) the non-linear weighting of the proximity Δ. Generally co-links in close proximity should have a higher score than co-links far apart. Hence, α needs to greater or equal to zero. Moreover, the higher α the closer the co-link proximity needs to be to have an effect on the final CPI score. This relation is illustrated by the plots in Figure 4.4. Due to different link usage in simple English and German Wikipedia, we expect the α-parameter to be different for each corpus.

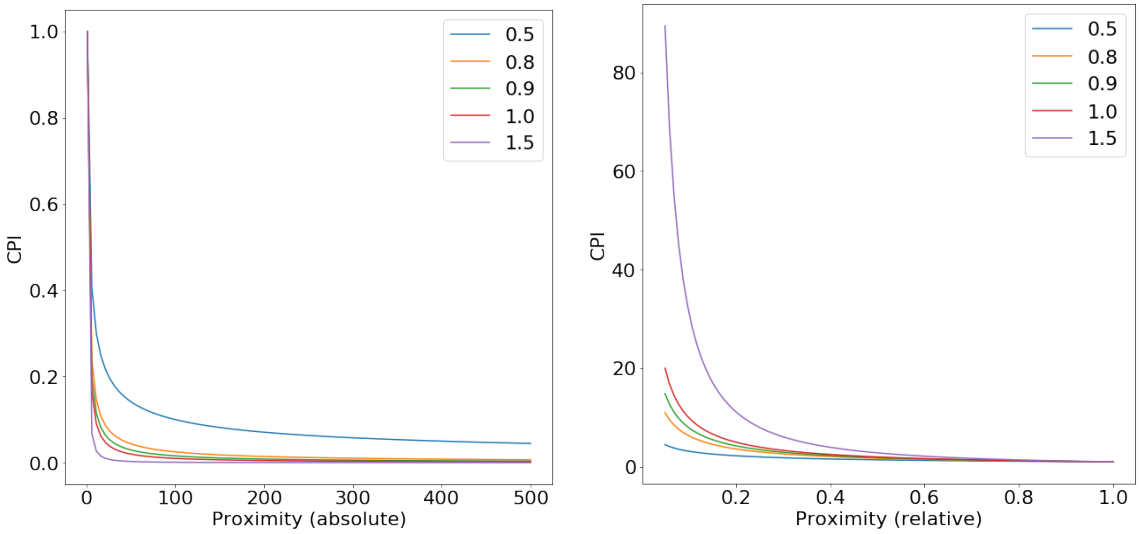

Figure 4.4: Effect of different α-values on absolute and relative proximity. The higher α the closer the proximity needs to be to affect final CPI score.

4.6.3 Inverse Link Frequency

The manual analysis of the previous study [39] revealed CPA's tendency to often recommend articles about years or political and geographic entities, because such article receive many in-links. The article "United States" was, for instance, the most recommended article while being also the article with the most in-links. Such articles are usually somehow relevant but for most recommendation scenarios too general. In order to give a preference to more specific recommendation and to penalize too general ones, we adapted the concept of Inverse Document Frequency from TF-IDF (Section 2.4) to the Inverse Link Frequency (ILF):

Table 4.2: ILF weighting schema derived from common IDF variants. $|D|$ is the number of documents in the corpus and n_a is the number of in-links of document a.

Weighting schema	**ILF weight**
Classic ILF	$log(\frac{\|D\|}{n_a})$
Smooth ILF	$log(\frac{1+\|D\|}{n_a})$
Probabilistic ILF	$log(\frac{\|D\|-n_a}{n_a})$
Okapi BM25 ILF [35]	$log(\frac{\|D\|-n_a+0.5}{n_a+0.5})$

The concept is already used in the field of citation analysis for scientific papers as CC-IDF citation-weighting schema [10]. However, Beel et al. [4] have shown that CC-IDF has about the same effectiveness as plain citation weight in a scientific paper recommender system. Yet, we test ILF as a logarithmic factor that is multiplied with the CPI score and depends on the number of in-links of the recommended article (and the total number of documents in the corpus). For the implementation of ILF we utilize common TF-IDF weighting schema (Table 2.2 and 4.2), where $|D|$ is the number of documents in the corpus and n_a is the number of in-links of document a.

We understand ILF as additional factor to the proximity Δ. Thus, we first conduct an offline evaluation to find the best proximity definition and then use the best proximity definition to find the best ILF factor in an additional evaluation.

4.7 Wikipedia Android App

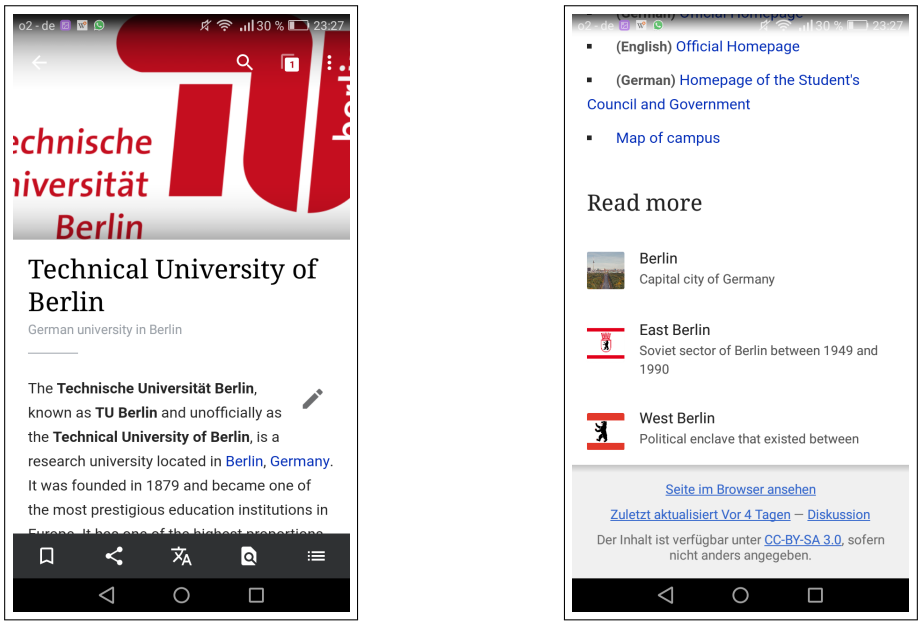

Figure 4.5: Wikipedia Android app. Article header (left). Recommendations are displayed in the "Read More" section at the bottom of each article (right).

The Wikipedia Android app is the front end, where we aim to serve the Citolytics recommendations. In fact, Wikipedia offers only in its mobile apps machine generated article recommendations. The actual Wikipedia website does not include a

recommender system. Aside from this difference, the content is identical in both the mobile apps and the website. Figure 4.5 shows the article view from within the Android app. The recommender system is located with the heading "Read more" at the bottom of each article page. By default three recommendations with their corresponding preview images are presented to the users. The display of preview images is problematic in terms of the evaluation: Images can significantly affect the click behavior. So we cannot rule out that an article is mainly clicked because of its image and not because of its content being relevant. Hence, click-oriented evaluations should be taken with precaution.

The actual evaluation mechanism from the Android app is developed as simple A/B test: Users of the Android applications are randomly assigned to two groups. While group A uses the old text-based recommender system with the *morelike:* query prefix and group B uses CPA recommendations with the *citolytics:* query prefix. Then, the user behavior (clicks and views) and the corresponding group assignment is track with the Wikipedia EventLogging system.

Recommendation Scenario When studying the effectiveness of recommender system, is it of importance to keep the reason, why the recommendation system is integrated into an application, in mind. Therefore, the scenario, in which the users of the Android app interact with the recommendations, should be defined. We define this scenario as follows:

The "Read more" feature, which is used to expose the recommendations to the users, is implicitly geared towards readers who are in "deep dive" mode. In other words, we assume that the users are reading for detail, not skimming or checking on a quick fact, when they scroll down to the very bottom of the page. Accordingly, we expected that a relevant recommendations will lead to further reading of the recommended article. With this regard we aim to evaluate the recommender system. This recommendation scenario goes along with the definition by Wikimedia Research[10].

4.8 Online Evaluation with Android App

After describing how recommendations are presented in the Wikipedia Android app, we define in this section the methodology of the online evaluation, which has the goal of comparing CPA and MLT based on user feedback from the app. We explain

[10] `https://meta.wikimedia.org/wiki/Research:Evaluating_RelatedArticles_recommendations`

how participants are recruited for the lab study and how user data is collected and evaluated. Moreover, we discuss the challenges of performing a real-world study.

4.8.1 Lab study

With a lab study we evaluated the performance of CPA and MLT, whereby we used our own test system and explicitly asked users to participate in our study.

Method To collect user feedback that then can use to evaluate the performance of CPA and MLT, we create a custom version of the Wikipedia Android app (Section 4.7). Our app version is identical to the original Wikipedia app except that in the back end the app is connected to our server not Wikipedia's. This allows us to compare the recommendations without requiring the Wikipedia production deployment. Recommendations are analyzed with an A/B testing procedure as it is common for this kind of research [12, 3]. One user group is exposed to CPA recommendations, while the other groups to MLT recommendations. Moreover, we can directly collect the user behavior data that is needed for our evaluation. With this evaluation approach we aim to compare the recommender systems in a quantitative manner. At the same time the lab study acts as proof of concept for a future real-world evaluation with the productive system. As explained in Section 2.2, we chose simple English and German as available languages.

EventLogging For the purpose of our lab study, we do not copy the whole EL infrastructure, since the limited number of users does not require high scalability. Instead, we configure a web server such that it accepts the EL requests and writes the event data in a log file on disk. In an additional step, the log files are then processed and written to a database which is used for evaluation. The complete lab study setup is bundled as Docker container and publicly available on GitHub[11].

Recruitment To recruit participants for the online evaluation, we created a single-page website[12], where we explained the background of the research and how to install the app. Additionally, we gave explicit instructions how to participate: We asked the users *"to install the app, browse five to ten articles and click on recommendations if there are any relevant"*. Then, the recruitment website was advertised through

[11] `http://github.com/mschwarzer/citolytics-docker`
[12] `http://bit.ly/citolytics`

various channels. It was sent to friends and colleagues, promoted on Twitter and related mailing lists. A screenshot of the website and the call-for-participation text can be found in the appendix. We set the goal to recruit at least 50 participants for each language version.

4.8.2 Real-world study

An online evaluation within a real-world application has the advantage that the user feedback is unlikely to be biased by the fact that users are aware of taking part in a study. Also, larger amounts of feedback can be collected. Yet, such study requires access to a real-world application. Therefore, we could not conduct a real-world study with the Wikipedia Android app, since the Wikimedia Foundation did not allow the deployment to their productive system. Even if the project had the support from Wikipedia developers and researchers, it did not receive the essential approval by the responsible product manager. The product manager argued that the deployment would cost additional resources while it's not clear whether it is beneficial. Moreover, recommender systems are not a high priority topic for Wikipedia.

For a non-profit organization such as Wikipedia the cost argument is crucial and cannot be easily encountered. On the other hand the problem that the implementation of a new feature such as an improved recommender system can never guarantee success is some kind of chicken-egg-problem, which probably most organizations face. A new feature can only be truly tested in a productive environment, i.e. via online evaluation. Offline evaluation and user-study can only provide hints about the performance of a new feature (Section 1.1). However, product managers expect evidence before deploying a new feature, since their task is it to ensure the quality of the system. As we heard from Wikipedia researchers this issue often determines their work. In general, we can only think about one solution: An additional testing or beta system that runs in parallel to the productive system and that allows the testing of new features before exposing them to the majority of the users. The testing system would be only available to a small group of users such that an unsuccessful feature would not risk the success of the whole website. A testing system like this currently does not exist at Wikipedia.

Discussion With this chapter we presented the steps that are needed to develop CPA from a research prototype to a production recommender system. Moreover, we derived improvements of the CPI model and technical requirements for the evalua-

tion. With the Wikipedia Android app we introduced our approach for evaluating the CPA-based recommender system in an online evaluation. Next, we present the results of our evaluation.

Chapter 5

Evaluation

In this chapter, we present the evaluation results of this thesis. We show the runtime optimization of the recommendation generation job, state the best parameters for the CPI model, analyze sample recommendations from MLT and CPA, and evaluate the user feedback collected in the online experiment.

5.1 Runtime Optimization

A key requirement for a productive system, especially if it has limited resources as Wikipedia has, is the runtime performance. On the contrary, the prototype from the preliminary study had only experimental character and, therefore, it was not optimized regarding the runtime. Hence, we rework the implementation of the CPA recommender system, whereby we put special focus on the runtime performance. In the following we describe which steps led to a significant decrease in runtime.

The bottleneck for distributed computing is usually the network. Sending data from on nodes to another takes a magnitude more time than moving data within a single node. Thus, minizing network traffic is generally worthwhile in distributed computing application. The Flink job for recommendation generation has the intermediate results from the link extraction step (Figure 4.2, step 1) that are in particular performance critical because they are sent over the network. The theoretical uncompressed output of this step consists of 37 billion records that are approx. 32 TB in size (for English Wikipedia).

In the preliminary implementation a *groupReduce* operator, which corresponds to the summation of the CPI formula (Equation 2.1), was applied on the mapper

Table 5.1: Results of runtime optimization. We achieved a significant decrease in runtime of Apache Flink recommendation generation job.

Job configuration	**Nodes**	**Avg. runtime**
No optimization (before thesis)	10	7h 45m
Optimization	8	2h 10m
Optimization (additional features)	5	3h 15m

output. But the *groupReduce* operator requires all grouped records to be available on one machine before its execution, which turned out to be a bottleneck. Hence, we replace the *groupReduce* with a simple *reduce* operator that allows execution even if not all group records are available. Therefore, the network traffic is minimized. Additionally, we specified the *CombineHint* of the *reduce* operator to be hash-based instead of sort-based, which was automatically suggested by the Flink optimizer.

Replacing the *coGroup* operator that was used for redirect resolution with a *leftOuterJoin* operator led to a further decrease in runtime (Figure 4.2, step 3). The redirect resolution was already in the preliminary implentation designed as *leftOuterJoin* operator. However, at the time of the first implementation the *leftOuterJoin* operator was not available in Flink and therefore a work-around with a *coGroup* operator was necessary.

Aside the optimization of the Flink abstractions, we also analyzed the second order functions with the Java Mission Control profiler. But we did not find any essential performance issues, since the second order functions are relatively simple, i.e. they use mainly Java primitives.

As result, these modifications led to a significant decrease in runtime. The average results of the runtime optimization are reported in Table 5.1. An additional *leftJoin* operator that is necessary for the computation of the Inverse Link Frequency factor (Section 5.2.2) is responsible for a longer runtime, i.e. it corresponds to the additional features (third row in Table 5.1). The experiment was performed on a cluster of 10 IBM Power 730 (8231-E2B) servers. Each machine had 2x3.7 GHz POWER7 processors with 6 cores (12 cores in total), 2 x 73.4 GB 15K RPM SAS SFF Disk Drive, 4 x 600 GB 10K RPM SAS SFF Disk Drive and 64 GB of RAM. Due to hardware failures our final performance optimizations could not be tested with all ten cluster nodes.

5.2 Offline Evaluation

In the following we present the results of our offline evaluation that aims to find the best CPI model parameters, whereby the evaluation is separated in two optimization steps. First different proximity definitions and α-values are tested. Subsequently, we determine the best Inverse Link Frequency schema based on the results from the first optimization step.

5.2.1 Proximity Definition & α-value

From our preliminary study with the English Wikipedia we already know that the best α-value lies between 0 and 2. Therefore, we compute our objective function for a series of α-values in this range and with absolute and relative proximity. Additionally, the objective function (Equation 2.7) is computed for structure proximity. To visualize the effect of the two investigated Wikipedia language, the objective function scores are displayed in Figure 5.1 for simple English in red and for German in blue.

Independent from the language, the best results are achieved with $\alpha = 0.9$ and absolute proximity. Moreover, we can assume the global maximum to be near $\alpha = 0.9$, since the plot shows a bell-shaped curve. For higher or lower α-values the objective function is decreasing. Regarding the proximity definition, absolute proximity outperforms relative and structure proximity. However, such an outcome was unexpected. It is astonishing that the best model parameter are identical for German and simple English. We would have expected that corpora of different sizes and different article properties (Section 2.2) lead to different model parameter, especially a different α-value. We must also note that $\alpha = 0.9$ is close to the best α-value from the previous study with the even bigger English corpus [39] where the optimized CPI model used $\alpha = 0.855$. It is as well unexpected that absolute proximity performs best, since relative proximity is convincingly theoretical justified with the different length of Wikipedia articles.

Nonetheless, best model parameters, which are independent from the corpus properties, are very much appreciated from a practical point of view. A recommender system that does not require optimization of model parameters is easier to use. In other words, it is out-of-the-box employable.

Aside the identical best model parameters, Figure 5.1 shows that CPA generally performs better with German than with simple English. This outcome, however, was expected. The German corpus is generally "bigger" than the simple English corpus. German has much more articles (2 million compared to 122,000 simple

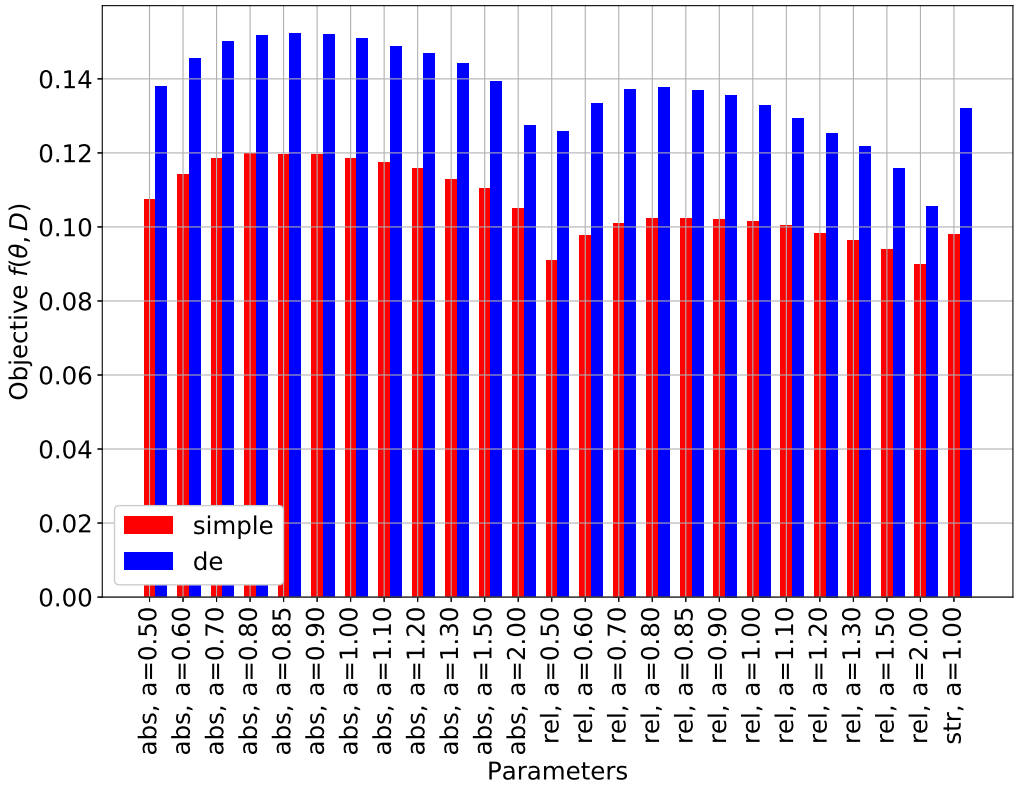

Figure 5.1: Objective function (offline performance) for simple English (red) and German (blue) with α-values and proximity definitions (abs=absolute, rel=relative, str=structure). Best result of the objective function is achieved with $\alpha = 0.9$ and absolute proximity.

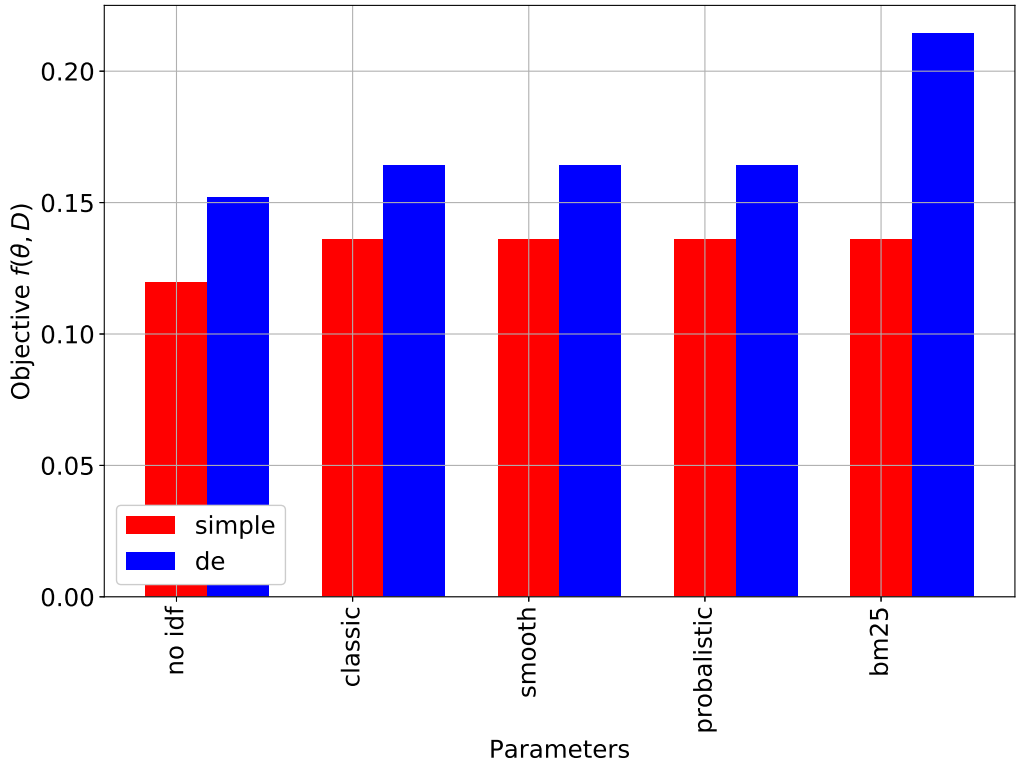

Figure 5.2: Objective function (offline performance) for simple English and German Wikipedia with different ILF weighting schema (no IDF=plain CPI). Best result of the objective function is achieved with Okapi BM25.

English articles) and more links connected the article with each other. Consequently, CPA has more information to work with and, therefore, is more likely to deliver better recommendations.

5.2.2 Inverse Link Frequency

To find out whether ILF improves the performance and which ILF schema performs best, we evaluate our objective function with the parameters discussed in Section 4.6.1. The results are shown in Figure 5.2.

In contrast to Beel et al. [4], the evaluation shows that ILF has a positive effect for our Wikipedia use case. All ILF weighting schemes perform better than the plain CPI, while the Okapi BM25 schema achieves the best result. Similar to Figure 5.1, the

result is again independent from the corpus. For both simple English and German, Okapi BM25 is the best ILF schema. Yet, we see that the relative performance difference among the ILF schema is stronger for German. We assume that this is caused by the wider distribution of in-links for German articles (Figure 2.2). In the German Wikipedia there are a magnitude more article without any in-links and at the same time the most linked article has ten times more links than the most linked article from simple English Wikipedia.

Discussion In conclusion, we derive from this evaluation the following equation as best and final CPI model:

$$\mathrm{CPI}(a,b) = \sum_{j=1}^{|D|} \Delta_j^{\text{absolute}}(a,b)^{-0.9} * log\left(\frac{|D| - n_a + 0.5}{n_a + 0.5}\right) \tag{5.1}$$

The final CPI is used for generating the recommendations for the manual analysis and the online evaluation (Section 4.8.1). Due to the use of the in-links n_a, the CPI is no longer bidirectional. Instead, CPI defines the ranking of article a being a recommendation for article b.

5.3 Sample Evaluation

Before evaluating user feedback on CPA and MLT recommendations, we first manually analyze recommendations of four sample articles for each language. The sample articles were chosen for their diversity and comprehensibility. The chosen articles were available in simple English and German:

The sample articles and their properties are presented in Table 5.2. "Technical University Berlin"[1,2] is the article about the authors home university. "Newspaper"[3,4] explains the frequently printed publication. "Brad Pitt"[5,6] is about the famous Hollywood actor. "Einstein field equations"[7,8] is a niche article on equations derived

[1] https://simple.wikipedia.org/wiki/Technical_University_of_Berlin
[2] https://de.wikipedia.org/wiki/Technische_Universität_Berlin
[3] https://simple.wikipedia.org/wiki/Newspaper
[4] https://de.wikipedia.org/wiki/Zeitung
[5] https://simple.wikipedia.org/wiki/Brad_Pitt
[6] https://de.wikipedia.org/wiki/Brad_Pitt
[7] https://simple.wikipedia.org/wiki/Einstein_field_equations
[8] https://de.wikipedia.org/wiki/Einsteinsche_Feldgleichungen

Article	Wiki	Size	Edits	Page views
Technical University of Berlin	simple	1,975	4	182
Technische Universität Berlin	de	55,702	24	65,136
Newspaper	simple	1,450	2	46,981
Zeitung	de	17,006	51	65,136
Brad Pitt	simple	4,104	11	4,949
Brad Pitt	de	28,845	12	688,346
Einstein field equations	simple	1,293	4	3,069
Einsteinsche Feldgleichungen	de	10,987	23	30,695

Table 5.2: Properties of sample articles from simple English (first row) and German (second row). Size is measured in character count of article content. Page views and edit count from November 2016 until October 2017.

from Albert Einstein's general theory of relativity. The top-3 recommendations sets of the four articles are shown in Tables 5.3 (simple English) and 5.4 (German).

In contrast to the manual analysis in [39], where we found obvious differences in the recommendations of CPA and MLT, here we hardly see any difference between the recommendations. In general, we consider most recommendations relevant, whereby both recommender systems perform better with German than simple English. For instance, you can possibly argue that the simple English recommendations "Cottbus" and "Bionics" for "Technical University Berlin" are irrelevant. On the other hand, we do not see any none irrelevant German recommendations. A problem we found in [39] was that MLT tends to recommend irrelevant niche articles and CPA tends to recommend too general articles. From the four samples we see that these issues has been resolved by our adapted version. Wikipedia's MLT modification of adding a popularity score removes niche articles, while the novel IFL factor of CPA penalizes too general articles. Looking at the simple English recommendations of "Brad Pitt", you can even argue that MLT with popularity score now recommends too general articles. While "Angelina Jolie" is a relevant recommendations, "Movie" and "1975" lack a direct connection to Brad Pitt. On the contrary, CPA also provides stable recommendations for "Brad Pitt" by recommending either actors who worked with him ("Casey Affleck" and "Eric Bana") or characters who he played ("Jesse James"). But given the fact that this issue is only visible in one of eight samples, we cannot

Rank	MLT recommendation	CPA recommendation
Seed	*Technical University of Berlin*	
1	Charlottenburg-Wilmersdorf	Darmstadt Uni. of Technology
2	Berlin	Bionics
3	Cottbus	Dresden University of Technology
Seed	*Newspaper*	
1	Magazine	Magazine
2	Movie	Editor
3	Book	Tabloid
Seed	*Brad Pitt*	
1	Angelina Jolie	Casey Affleck
2	Movie	Jesse James
3	1975	Eric Bana
Seed	*Einstein field equations*	
1	Sphere	Momentum
2	Photon	Matter
3	Maxwell's equations	Energy

Table 5.3: Sample recommendations from simple English by MoreLikeThis (left) and Co-Citation Proximity Analysis (right).

state that this is a general problem of MLT with popularity score.

Discussion All in all, the manual analysis showed that both recommender systems perform generally well and the tiny differences that we found cannot be generalized. Hence, we cannot derive from this analysis whether CPA or MLT is superior.

5.4 Online Evaluation

After optimizing CPI in an offline evaluation, we present in this section the results of our online evaluation, which has been conducted with the Wikipedia Android app in a lab study.

Rank	MLT recommendation	CPA recommendation
Seed	*Technische Universität Berlin*	
1	Technische Universität	Freie Universität Berlin
2	Technische Universität Dresden	Technische Universität München
3	Technische Uni. Braunschweig	Berlin
Seed	*Zeitung*	
1	Tageszeitung	Zeitschrift
2	The Independent	Buch
3	Stuttgarter Zeitung	Fernsehen
Seed	*Brad Pitt*	
1	Angelina Jolie	Angelina Jonie
2	Oscarverleihung 2012	George Clooney
3	Michael Fassbender	Die Kunst zu gewinnen - Moneyball
Seed	*Einsteinsche Feldgleichungen*	
1	Allgemeine Relativitätstheorie	Allgemeine Relativitätstheorie
2	Raumzeit	Vakuumlösung
3	Vierervektor	Kosmologische Konstante

Table 5.4: Sample recommendations from German by MoreLikeThis (left) and Co-Citation Proximity Analysis (right).

5.4.1 Participants

Participants for our lab study have been mainly recruited through our lab study website. The experiment data has been collected in the period from Mid of August until beginning of October 2017. Yet, the number of participants turned out to be far below our expectations. Even if the website attracted visitors from various countries with the majority coming from Germany and Japan, we achieved in total only 114 views and 79 visitors (Table 5.5). For web tracking we rely on Google Analytics[9].

In total 33 users installed our app (simple English: 15 users, German: 18 users) resulting in an install-rate of 41.7%.

[9] `http://www.google.com/analytics`

Table 5.5: Number of views and visitors by country on lab study website. Top 10 countries are displayed. In total 114 views and 79 visitors.

Country	Views	Visitors
Germany	57	39
Japan	39	25
Austira	2	2
Switzerland	2	1
United Kingdom	2	1
Indonesia	2	2
Italy	2	2
Thailand	2	2
Hong Kong	1	1
India	1	1

5.4.2 Lab study results

With a total number of 33 participants we did not achieve a statistically significant outcome that prove that either CPA or MLT is the superior recommender system in terms of clicks. Also, regarding the user behavior we could not find any significant difference between CPA and MLT.

The majority of events has been collected in the first third of the study time range (Figure 5.3). This can be explained by that most users directly responded to our call for participation by installing the app and using it once or twice. Only a few users used the app more than two times. One reason for this might be that the app was promoted as research subject. So the users did not adopt the app as default way to access Wikipedia content because they were aware of its research purpose.

Aside the app usage, the overall performance of CPA and MLT is of more importance (Table 5.6). Surprisingly, both recommender systems have independent from the language nearly the exact same CTR (approx. 12%). While the absolute number of views and clicks differ. German CPA is used most (648 views and 83 clicks). German MLT has the lowest number of interactions (198 views and 25 clicks). For simple English the views and clicks are on the same level (approx. 300 views and 38 clicks). As a result the CTR of CPA and MLT for simple English and German show no difference that is statistically significant at level of 5% with $p_{SimpleEnglish} = 0.8163$ and $p_{German} = 0.9594$.

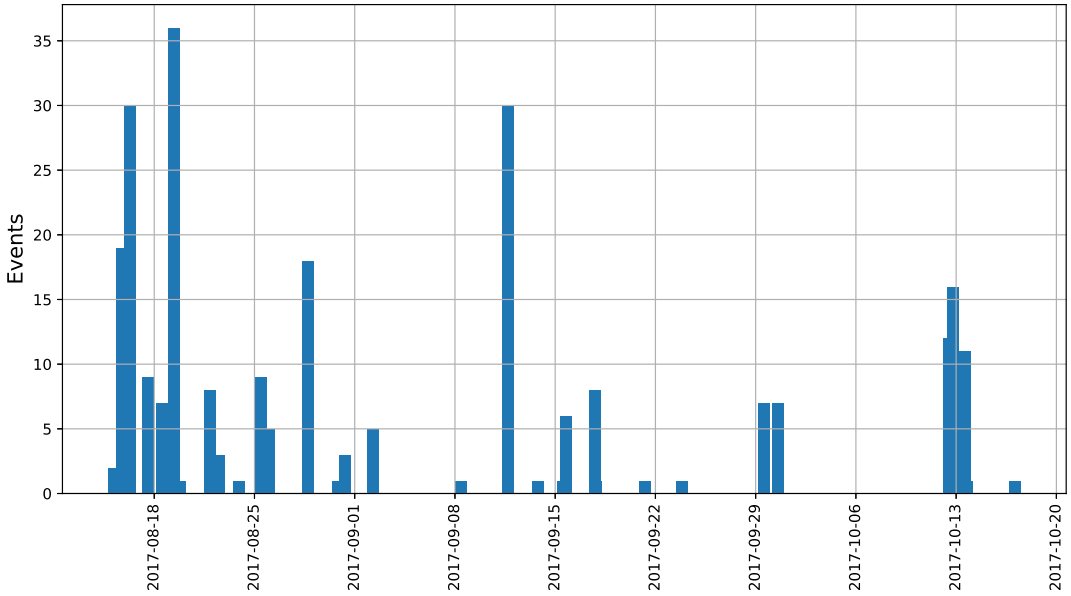

Figure 5.3: Number of collected events (user-study data points) from Android app (simple English) during the user-study from August to October 2017.

For Long Clicks and Long CTR we see a similar outcome. Despite that for German there is an advantage for CPA with a Long CTR of 6.02%, whereas MLT has a Long CTR of 3.03%. However, this difference is also not statistically significant. On the other hand, the average time spent on an recommended article differs largely for simple English (CPA 26s and MLT 46s), whereas for German the time spent the difference is little (CPA 35s and MLT 32s).

We can state that a CTR of 12% is high but realistic. The recommendations are displayed at the very bottom of the article page (Section 4.7) and, therefore, from a lower CTR would be expected from the user interface point of view. Yet, we assume the click rate to be realistic, because Wikipedia Research observed with 16% a similar high CTR in a similar experiment[10].

Despite the click rate, we assume that our call for participation, in which we explicitly ask to click on the recommendations, led to this unrealistic reading behavior. The measurements of time spent and max. viewed are too high: An average max. viewed of 99% is not realistic, especially when also taking the avg. time spent of 26 seconds into account. A user simply does not read (or scan) a whole Wikipedia

[10] `https://www.mediawiki.org/wiki/Reading/Web/Projects/Related_pages`

article in 26 seconds on average.

Table 5.6: Overall performance measures per recommender system. CPA and MLT perform similarly well. Performs show no statistical significant difference.

	simple English		***German***	
	CPA	**MLT**	**CPA**	**MLT**
App Installs	15		18	
Views	288	318	648	198
Clicks	37	39	83	25
CTR (%)	12.8	12.3	12.8	12.6
Long Clicks	16	19	39	6
Long CTR (%)	5.5	6.0	6.0	3.0
avg. Time Spent (s)	26.1	46.0	35.9	32.3
avg. Max. Viewed (%)	99.8	97.8	98.3	92.3

The overall performance measures do not reveal a difference in CPA and MLT. For more insights, we continue with a look at the most recommended (viewed) or clicked recommendations (Table 5.7 and 5.8). As expected, "United States" is the most recommended article for simple English CPA. We expected "United States" to be in the top-5, because CPA's tendency to favor articles with a high number of in-links and "United States" is the article with most in-links. However, it is surprising that "United States" is with distance the most recommended article and that the tendency is visible at such a small sample size. Given that the introduction of the IFL factor was supposed to reduce the in-link bias, this outcome is even more suprising.

Table 5.7: Top-5 viewed recommendations per source. (# = views)

	simple English				***German***			
	CPA	#	**MLT**	#	**CPA**	#	**MLT**	#
1	Unitied States	14	Airship	7	Mango	8	Gasoline	8
2	God	9	Refraction	6	Mais	8	Obersee (Bodensee)	7
3	Moses	9	Telescope	6	Mustererkennung	6	Alternative fuel vehicle	6
4	Jesus	8	Judaism	5	Bohne	6	Munchausen syndrome	5
5	United Kingdom	7	Singing	5	Weizen	6	Österreichischer Rundfunk	4

Moreover, it is also notable that religious (simple English: God, Moses, Jesus) and food-related articles (German: Mango, Mais, Bohne, Weizen) are frequently

recommended. We assume that this outcome can be explained by single users who extensively browsed articles from a particular topic and the small sample size that did not balance the effect of those single users. With a larger sample size we would expect more diverse topics or topics that are generally popular among Wikipedia users like celebrities[11]. The recommendations by MLT do not show any kind of bias. We cannot state whether this is because of an conceptional advantage or just random.

When analyzing the most clicked recommendations, the small sample size problem is even more obvious. None of the most clicked recommendations has more than two clicks. Giving a statement on any differences of CPA and MLT is with this small sample impossible.

Table 5.8: Top-5 clicked recommendation per source. (# = clicks)

	simple English				***German***			
	CPA	#	**MLT**	#	**CPA**	#	**MLT**	#
1	United States	2	Brandenburg Gate	2	Mais	2	Bodensee	2
2	Torah	2	Photography	2	Bohne	2	Überlinger See	2
3	London	2	Soprano	1	Berlin	2	ORF 2	1
4	The Guardian	2	Bible	1	Hamburger SV	1	Gas station	1
5	Old Testament	1	Montpeller, Indiana	1	Deutschland	1	Deutscher Handballbund	1

An often discussed design decision of recommender systems is the specification of k, i.e. how many recommendations should be presented to the user. Wikipedia chose to provide the top-3 recommendations in the Android app. The design decision is driven by the assumption that user attention decreases as the rank of a recommendation decreases because the user's eye movement is usually from top to bottom. Put differently, the recommender system should rank the most relevant recommendation at the top. This expectation is reflected by rank-based performance measures like MAP (Section 2.9).

Consequently, we would expect that in our data we find top recommendations receive the most clicks (Table 5.9). However, this is not the case. Independent from recommender system or Wikipedia language, number of clicks does not decrease with decreasing rank. With simple English, for instance, the most clicks are received by the third not the first ranked recommendations. For MLT's simple English recommendation the outcome is even the opposite of what would be generally expected. The clicks increase as the rank decreases. Having such an outcome, brings up the question whether the use MAP as performance measure is valid for this use case or if

[11] `https://en.wikipedia.org/wiki/Wikipedia:Top_25_Report`

Table 5.9: Clicks per recommendation rank and source.

	simple English		***German***	
	CPA	**MLT**	**CPA**	**MLT**
1	11	11	33	11
2	8	13	19	11
3	18	15	31	3

rank-independent measures such as CTR are more likely to be suitable. With regard to the small sample size, we cannot provide an answer to this question. Nonetheless, we hypothesize that the low value of $k = 3$ is responsible for this outcome and that a higher value of k would lead to different outcome.

Due to the low number of participants, a more detailed evaluation of article properties (length, topics, etc.) that was planned for the real-world study has not been performed with the lab study data.

Discussion Given these evaluation results, we can state that CPA and MLT are both similar well suited as Wikipedia recommender system. Also, the similar performance for simple English and German shows that the corpus does not affect the outcome of the comparison. A similar results can be therefore expected for other Wikipedia languages. Furthermore, the result of the online evaluation does not only allow a statement on CPA and MLT, it also shows the validity of the offline evaluation with "See also" links and click streams. Both offline and online evaluation led to the same result. With the help of the quasi-gold standards the offline evaluation predicted the results of the online evaluation.

Chapter 6

Conclusion

In this chapter, we summarize our findings, draw conclusions and propose areas of future work.

6.1 Summary

The contribution of this work consists of mainly two parts: First, the development of CPA from a research prototype to a large-scale production Wikipedia recommender system. The development includes an increased efficiency of the Apache Flink job, the integration into Wikipedia's back and front end, and the conceptional improvement of the CPI model. Second, the comparison of CPA and MLT with a manual analysis and an online evaluation that is based on real user feedback.

Development of a Large-Scale Recommender System One major element of the implementation work was the development of an efficient Apache Flink job, which generates the recommendations from a Wikipedia XML dump. The job was optimized with respect to the runtime performance and scalability. To be precise, we achieved a decrease in runtime from 4:45h to only 2:10h while consuming less computing power. At the same time, we added additional features that now allow a more complex CPI computation, e.g. including IFL factor. These modifications increase the efficiency and quality of the link-based recommendations. For the deployment as Wikipedia recommender system, several components of the Wikipedia infrastructure have been modified. We integrated the link-based recommendations into Wikipedia's MediaWiki/CirrusSearch API in such way that recommendations

are loaded from Elasticsearch and made accessible to the Android app. We added to the Android app an A/B testing mechanism that shows either CPA- or MLT-based recommendations to randomly assigned user groups. For the frequent and automatic generation of recommendations we added a procedure to Wikipedia's existing Apache Oozie setup. The Oozie procedure runs the Apache Flink job in a YARN environment and populates the results into Elasticsearch. In addition, we developed an approach to read user data, which Wikipedia's EventLogging system collects, and to analyze it in an separate database. As a result of this work, the CPA-based recommender system is capable of being deployed to Wikipedia's production environment. The developed system is scalable to provide recommendations to several million Wikipedia users. Moreover, it would allow a large-scale online evaluation. Parts of our development are already presented in a separate publication [38].

Evaluation of a Large-Scale Recommender System Secondly, we contributed the evaluation of the link-based CPA as Wikipedia recommender system in comparison to the text-based MLT. With offline evaluations we tested three different methods of quantifying the co-link proximity and found that the plain number of words between links (absolute proximity) works best. We also showed that the introduction of an Inverse Link Frequency (ILF) factor leads to a further improvement of CPA. In particular, the IFL factor, which we derived from the concept of TF-IDF, decreases CPA's tendency of recommending too general articles. For the offline evaluation we used Wikipedia's click stream and "See also" links data as quasi-gold standards, whereby we added multi language support to increase each gold standard's coverage. In addition, we found that recommendations generated from edit logs are not suitable as gold standard for evaluating a Wikipedia recommender system.

The main objective of this thesis was the evaluation of CPA and MLT in an online scenario. In particular, we used the Wikipedia Android app for the evaluation. We performed a lab study with 33 participants in total. The study's outcome presented no significant difference of CPA and MLT. Both recommender systems performed similarly well with respect to our performance measures. The metrics CTR and Long CTR, but also time spent and percentage viewed, have been used to assess the recommendations. Given that we evaluated the user behavior in a lab study, we consider our findings as representative, because a similar study conducted by Wikipedia Research resulted in a similar click rate on recommendations. Furthermore, our evaluation proves the language independence of CPA, since similar results for simple English and German have been achieved. Despite the successful lab study,

the real-world online evaluation with the Wikipedia audience could not be conducted as part of this thesis due to missing support from the Wikimedia Foundation.

Aside the offline and online evaluation, we as well analyzed manually recommendations samples of CPA and MLT. In the manual analysis we did not find any major differences of CPA and MLT, too. Thus, these results are in accordance with the online evaluation. Additionally, we can state from the samples that the IFL factor for CPA and the popularity factor, which has been added to MLT by Wikipedia, solve issues of both recommender system that we address in our first study [39].

6.2 Conclusion

We started this thesis with the question whether CPA is suitable of being deployed to a large-scale production environment such as Wikipedia. Given our implementation work and the feedback from Wikipedia's developers, we can confirm CPA's suitability. Even if the final deployment to Wikipedia's production system failed due to lack of support by Wikipedia's product management team, we prove CPA's practical use in a test system setup that was used in our lab study.

In preparation for the online evaluation, we developed and tested the IFL factor for CPA. The ILF factor increased the recommendation quality without using any additional data sources. ILF uses only links and is, therefore, only content-based. On the contrary, the popularity score improvement of MLT requires information on the number of clicks on an article. Put differently, Wikipedia's MLT version is in contrast to the original MLT implementation not only text-based (content-based), instead it relies as well on user data. Thus, Wikipedia's MLT suffers like other user-based recommender systems from the cold start problem. The cold standard problem refers to the fact that at the first start of the system, when no user data is available, the recommender system cannot work properly. Transfered to Wikipedia, at first enough user data needs to be collected to compute popularity scores. Even if the ILF factor represents an improvement of CPA for the Wikipedia use case, it is questionable if IFL has the same effect on other document types. We doubt that the concept can be directly transfered to CPA's original domain, scientific publications, because Wikipedia articles tend to have generally more in-links than scientific publications have citations. A different weighting schema is probably required.

In the evaluation part of this thesis, the objective was to compare CPA and MLT in an online evaluation. From the outcome of the lab study we cannot derive a statement on the superiority of CPA or MLT. Both recommender systems perform

equivalent. Thus, the corresponding research questions remains unanswered. However, CPA and MLT are conceptional different even if the online evaluation did not reveal a significant difference. From the few samples of the manual analysis we observed already tiny differences. Therefore, we argue that the identical performance is merely caused by the low number of participants. If more data would have been collected in the lab study, the differences between CPA and MLT would be visible. An other reason for the indifference might be that the data does not originate from a real-world application, i.e. the users were aware of taking part in a study. Hence, we cannot give a final answer to our research question that was whether CPA or MLT performs better. But we expect a future large-scale online evaluation to answer this question. However, we can draw additional conclusions: The results for simple English and German are similar. The Wikipedia language does not affect the outcome. Other languages would most probably result in similar outcomes. Consequently, our findings are generalizable if the corpus consists of similar with links interconnected articles. In addition, the outcome of the online evaluation also shows the validity of the offline evaluation. Given that the offline evaluation predicted the online evaluation, we no longer need to refer to "See also" links and click streams as quasi-gold standards. The online evaluation proved their use. Thus, we declare "See also" links and click streams as valid gold standards.

6.3 Future Work

The development and evaluation of CPA as Wikipedia recommender system leads to additional research questions that we propose in the following. We distinguish the future work in two areas: Evaluation and further improvements of CPA.

6.3.1 Evaluation

The evaluation part of the thesis did not reveal any significant difference of CPA and MLT. So the question of which recommender system is superior remains. Hence, in the future additional evaluation work is needed. CPA is ready to be deployed at Wikipedia. Thus, it is feasible to conduct a real-world study in the near future. Furthermore, the possible outcome of a large-scale online evaluation is promising in multiple ways. On one side, the real-world study is expected to deliver a judgment on the performance of the two recommender systems. With a larger amount of evaluation data differences in the performance should easier to identify. On the

other side, the collected evaluation data could be used by other researchers, because the data does not come from a private owned live system. If Wikipedia does not allow a deployment, we also see other websites that run with the MediaWiki software as potential platforms for an online evaluation. But integrating CPA into another infrastructure would require additional work. Moreover, another website, especially if it is private owned, would not offer the same level of openness and reproducibility as Wikipedia. Therefore, websites that rely on the same technologies as Wikipedia and have similar encyclopedic content with interconnecting links are most qualified for such an approach.

Aside the online evaluation, a qualitative user study could also provide detailed insights on CPA and MLT. In questionnaires or interviews subjects can be ask to identify the conceptional differences of the CPA and MLT recommendations. Findings of such study would also allow a further improvement of the recommender systems.

6.3.2 CPA Improvements

In this work we successfully showed how to improve CPA. However, we propose to further extend the idea of CPA.

CPA uses co-links or co-citations. But the concept can be also transfered to other domains or to more abstract relations. For instance, CPA could be also applied on references in patent descriptions to find related patents. Or if CPA's ideas is understood in a more abstract or broader sense, relations could be also extracted using words or entities. In Wikipedia, for example, usually only the first appearance of a term is linked to its article. To get more co-links, a preprocessing step could add links to all the later appearances of linked terms. As we saw by comparing simple English and German, more links lead to better results. Therefore, we assume that creating additional links would improve CPA. Likewise, an approach like this can be also applied on named entities that are not connected to any article in the corpus. As a result, CPA would be also useful for non-encyclopedic corpora like news content, which often does not contain interconnecting links.

Lastly, we suggest to redesign CPA in such way that it uses streaming instead of batch processing. Right now, the whole Wikipedia dump needs to be processed to provide updated recommendations when Wikipedia's content changes. This creates an overhead, which should be avoided in a production environment. Given a data set that contains only the changes, CPA recommendations can be computed with a more efficient streaming approach if the implementation is adjusted accordingly.

Bibliography

[1] Citation Proximity Analysis (CPA)-A new approach for identifying related work based on Co-Citation Analysis. *Proc. of the 12th Int. Conf. on Scientometrics and Informetrics (ISSI'09)*, 2, 2009.

[2] Alexander Alexandrov, Rico Bergmann, Stephan Ewen, Johann-Christoph Freytag, Fabian Hueske, Arvid Heise, Odej Kao, Marcus Leich, Ulf Leser, Volker Markl, et al. The stratosphere platform for big data analytics. *The VLDB Journal*, 23(6):939–964, 2014.

[3] Riccardo Bambini, Paolo Cremonesi, and Roberto Turrin. A Recommender System for an IPTV Service Provider: a Real Large-Scale Production Environment. *Recommender Systems Handbook*, (March):299–332, 2011.

[4] Joeran Beel, Corinna Breitinger, and Stefan Langer. Evaluating the CC-IDF citation-weighting scheme: How effectively can 'Inverse Document Frequency' (IDF) be applied to references? *Proceedings of the iConference 2017*, pages 1–11, 2017.

[5] Joeran Beel, Corinna Breitinger, Stefan Langer, Andreas Lommatzsch, and Bela Gipp. Towards reproducibility in recommender-systems research. *User Modeling and User-Adapted Interaction (UMAI)*, 26, 2016.

[6] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. Research-paper recommender systems: a literature survey. *Int. Journal on Digital Libraries*, 2015.

[7] Joeran Beel and Stefan Langer. A Comparison of Offline Evaluations, Online Evaluations, and User Studies in the Context of Research-Paper Recommender Systems. *TPDL*, 2015.

[8] Joran Beel, Stefan Langer, Marcel Genzmehr, and Andreas Nurnberger. Introducing docear's research paper recommender system. In *JCDL*, 2013.

[9] James Bennett and Stan Lanning. The Netflix Prize. *KDD Cup and Workshop*, pages 3–6, 2007.

[10] Kurt D. Bollacker, Steve Lawrence, and C Lee Giles. CiteSeer : An Autonomous Web Agent for Automatic Retrieval and Identification of Interesting Publications. *Proceedings of the 2nd International Conference on Autonomous Agents*, pages 116–123, 1998.

[11] Paris Carbone, Stephan Ewen, Seif Haridi, Asterios Katsifodimos, Volker Markl, and Kostas Tzoumas. Apache flink: Stream and batch processing in a single engine. *Data Engineering*, page 28, 2015.

[12] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. The youtube video recommendation system. In *RecSys*, 2010.

[13] Simon Dooms, Toon De Pessemier, and Luc Martens. A User-centric Evaluation of Recommender Algorithms for an Event Recommendation System. *RecSys*, 2011.

[14] Michael D. Ekstrand, Michael Ludwig, Joseph A. Konstan, and John T. Riedl. Rethinking the Recommender Research Ecosystem : Categories and Subject Descriptors. *RecSys '11*, pages 133–140, 2011.

[15] Masaki Eto. Evaluations of context-based co-citation searching. *Scientometrics*, 94(2), 2013.

[16] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proc. of the 4th ACM conference on Recommender systems*, pages 257–260. ACM, 2010.

[17] Bela Gipp. *Citation-based plagiarism detection: Detecting disguised and cross-language plagiarism using citation pattern analysis*. 2014.

[18] Asela Gunawardana and Guy Shani. A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. 10:2935–2962, 2009.

[19] Mohammad Islam, Angelo K Huang, Mohamed Battisha, Michelle Chiang, Santhosh Srinivasan, Craig Peters, Andreas Neumann, and Alejandro Abdelnur. Oozie: towards a scalable workflow management system for hadoop. In *Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies*, page 4. ACM, 2012.

[20] Karen Sparck Jones. Index term weighting. *Information Storage and Retrieval*, 9(11), 1973.

[21] Petr Knoth and Anita Khadka. Can we do better than co-citations? - bringing citation proximity analysis from idea to practice in research article recommendation. In *BIRNDL@SIGIR*, 2017.

[22] Roy Lachica, Dino Karabeg, and Sasha Rudan. Quality, Relevance and Importance in Information Retrieval with Fuzzy Semantic Networks. *Proc. TMRA*, 2008.

[23] Jimmy Lin and W John Wilbur. PubMed related articles: a probabilistic topic-based model for content similarity. *BMC bioinformatics*, 8:423, jan 2007.

[24] Shengbo Liu and Chaomei Chen. The proximity of co-citation. *Scientometrics*, 91(2):495–511, 2012.

[25] Christopher D. Manning and Prabhakar Raghavan. *An Introduction to Information Retrieval*. Cambridge University Press, 2009.

[26] Iv Marshakova. System of document connections based on references. *Scientific and Technical Information Serial of VINITI*, 6, 1973.

[27] John Martyn. Bibliographic Coupling. *Journal of Documentation*, 20(4):236–236, apr 1964.

[28] Michael McCandless, Erik Hatcher, and Otis Gospodnetic. *Lucene in Action: Covers Apache Lucene 3.0*. Manning Publications Co., 2010.

[29] Brian McFee, Thierry Bertin-Mahieux, Daniel P.W. Ellis, and Gert R.G. Lanckriet. The Million Song Dataset Challenge. In *Proceedings of the 21st international conference companion on World Wide Web - WWW '12 Companion*, pages 909–916, New York, New York, USA, 2012. ACM Press.

[30] Sean M McNee, Istvan Albert, Dan Cosley, Prateep Gopalkrishnan, Shyong K Lam, Al Mamunur Rashid, Joseph A Konstan, and John Riedl. On the Recommending of Citations for Research Papers. In *Proc. of the 2002 ACM Conf. on Computer Supported Cooperative Work*, 2002.

[31] Yann Ollivier and Pierre Senellart. Finding Related Pages Using Green Measures : An Illustration with Wikipedia. *Proceedings of AAAI*, pages 1427–1433, 2007.

[32] Jessica Perrie, Yanqi Hao, Zack Hayat, Recep Colak, Kelly Lyons, Shankar Vembu, and Sam Molyneux. Implementing Recommendation Algorithms in a Large-Scale Biomedical Science Knowledge Base. *CoRR*, abs/1710.0:1–21, oct 2017.

[33] Pearl Pu, Li Chen, and Rong Hu. A user-centric evaluation framework for recommender systems. In *Proc. of the fifth ACM conference on Recommender systems*, pages 157–164. ACM, 2011.

[34] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation*, 60(5):503–520, 2004.

[35] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. Okapi at trec-3. *Nist Special Publication Sp*, 109:109, 1995.

[36] G Salton, A Wong, and CS Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18, 1975.

[37] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[38] Malte Schwarzer, Corinna Breitinger, Moritz Schubotz, Norman Meuschke, and Bela Gipp. Citolytics - a link -based recommender system for wikipedia. In t.b.d, editor, *Proceedings of the 11th ACM Conference on Recommender systems (RecSys 2017)*, 8 2017.

[39] Malte Schwarzer, Moritz Schubotz, Norman Meuschke, and Corinna Breitinger. Evaluating Link-based Recommendations for Wikipedia. *Proceedings of the 16th ACM/IEEE Joint Conference on Digital Libraries (JCDL'16)*, pages 191–200, 2016.

[40] James G. Shanahan and Laing Dai. Large scale distributed data science using apache spark. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 2323–2324, New York, NY, USA, 2015. ACM.

[41] Guy Shani and Asela Gunawardana. Evaluating Recommendation Systems. In *Recommender Systems Handbook*, pages 257–297. Springer US, Boston, MA, 2011.

[42] Henry Small. A New Measure of the Relationship Two Documents. *Journal of the American Society for Information Science*, 24, 1973.

[43] Koen Smets, Bart Goethals, and Brigitte Verdonk. Automatic Vandalism Detection in Wikipedia : Towards a Machine Learning Approach. *Proceedings of AAAI 2008 Workshop on Wikipedia and Artificial Intelligence An Evolving Synergy WikiAI08 (2008)*, pages 43–48, 2008.

[44] Allison Woodruff, Rich Gossweiler, James Pitkow, Ed H Chi, and Stuart K Card. Enhancing a Digital Book with a Reading Recommender. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*. ACM, 2000.

[45] Ellery Wulczyn and Dario Taraborelli. Wikipedia Clickstream. *Figshare*, 2017.

[46] Hua Zheng, Dong Wang, Qi Zhang, Hang Li, and Tinghao Yang. Do clicks measure recommendation relevancy? an empirical user study. In *Proc. of the 4th ACM conference on Recommender systems*, pages 249–252. ACM, 2010.

Appendices

Appendix A: Abbreviations

e.g.	exempli gratia
i.e.	id est
I/O	Input / Output
CoCit	Co-Citation
CPA	Co-Citation Proximity Analysis
CPI	Co-Citation Proximity Index
CSV	Comma-separated Values
CTR	Click-Trough-Rate
ES	Elasticsearch
EL	EventLogging
HDFS	Hadoop Distributed File System
IR	Information Retrieval
MAP	Mean Average Precision
MLT	MoreLikeThis
TF-IDF	Term Frequency - Inverse Document Frequency
VSM	Vector Space Model
RS	Recommender System
XML	Extensible Markup Language

Appendix B: EventLogging Schema

The user data for our evaluation has been collected with Wikipedia's EventLogging system (Section 4.5). The EventLogging system works with pre-defined schema that define which data is collected. These logging schema are presented in the following tables:

MobileWikiAppArticleSuggestions *Track when the user views or interacts with the "read more" suggestions at the bottom of the current article.*	
action (String)	Which user action triggered this event: "shown" is used when the,suggestions are first shown to the user, and "clicked" is used when the,user clicks one of the suggestions.
appInstallID (String)	AppInstallID that's unique to each app install and is used to track user across this and other features in the mobile app
pageTitle (String)	Title of the page currently being viewed
readMoreList (String)	List of page suggestions displayed to the user (separated by a pipe symbol)
readMoreIndex (Int)	Index of the suggested page that was clicked (only used with action=clicked)
readMoreSource (Int)	The source of the Read More suggestion list.

MobileWikiAppSessions *Track user behavior during a session of using the app.*	
length (Integer)	The length of this session, in seconds.
appInstallID (String)	AppInstallID that's unique to each app install and is used to track user across this and other features in the mobile app
totalPages (Int)	Number of total pages viewed during this session.
fromSearch (Int)	Number of pages viewed that came from the Search bar.
fromInternal (Int)	Number of pages viewed that came from tapping regular internal links on a page.
fromExternal (Int)	Number of pages viewed that came from external links (e.g. from the Browser app).

MobileWikiAppPageScroll *Measure how much of the page the user scrolls through.*	
pageID (Integer)	The ID of the article to which this event applies.
timeSpent (Integer)	Amount of time, in seconds, that the user spent reading this page, before leaving the page for any reason (going to another page, another part of the app, leaving the app, etc.)
pageHeight (Integer)	The total height of the page, in device-independent pixels.
maxPercentViewed (Integer)	The total percent of the page height that was reached.
appInstallID (String)	AppInstallID that's unique to each app install and is used to track user across this and other features in the mobile app

Appendix C: User Study Website

The following text has been used to promote the participation in our online evaluation:

> Dear all,
>
> We are conducting a user-study as part of our research on recommender

systems. The goal of this study is to evaluate the performance of a novel Wikipedia recommender system. For that reason we provide a custom version of the Wikipedia Android app for two languages (simple English and German). The app is identical to the original Wikipedia app expect that article recommendations, which can be found at the bottom of each article in the "Read more" section, are generated differently. To be precise, the novel approach generates recommendations based on links, whereas the current approach used by Wikipedia is text-based. For evaluation of the recommender system we collect data about the user behavior of each participant. In order to collect sufficient data we kindly ask you to install the app, browse five to ten articles (use "Explore" feature if you cannot think about any) and click on recommendations if there are any relevant.

The Android app is available on the following website:

`https://mschwarzer.github.io/citolytics-demo/userstudy/`

Your effort: 15 minutes.

Procedure:

- Download and install Android app

- Browse five to ten articles (use "Explore" feature if you cannot think about any)

- Click on recommendations if there are any relevant

Requirements:

- Android phone

Your participation will be a valuable addition to our research and findings could lead to better recommendations in Wikipedia or other websites.

Thank you in advance.

Best regards,

Malte Schwarzer

A screenshot of website where the our Wikipedia Android app could be downloaded is shown in Figure 6.1.

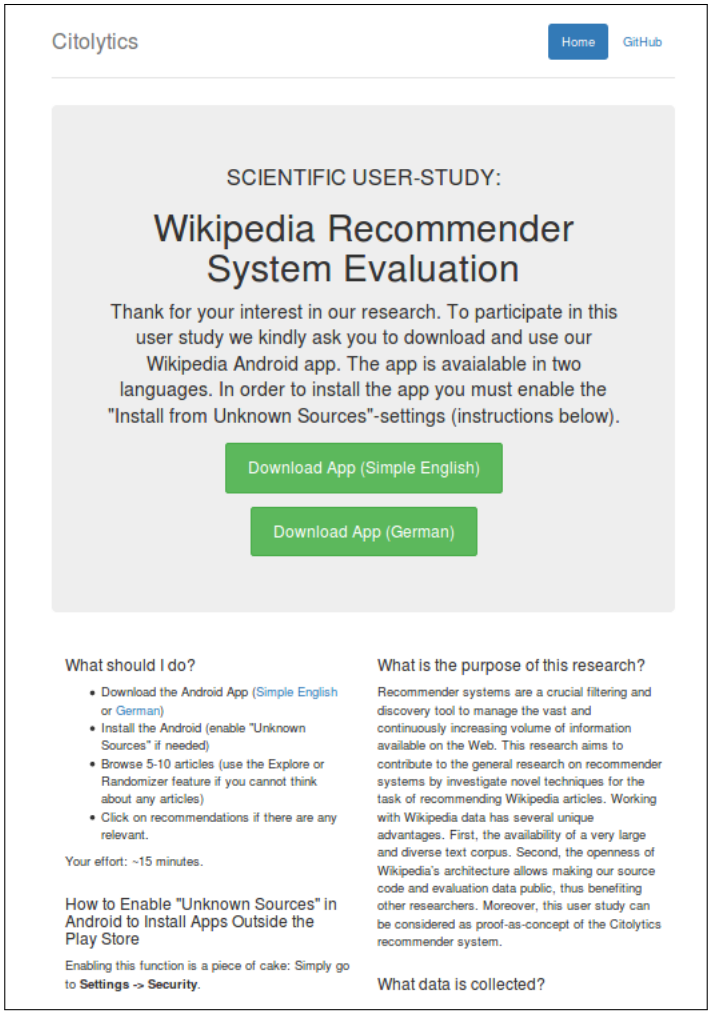

Figure 6.1: Screenshot of the user study website.