Dissertation

Leveraging Recommender Systems for the Creation and Maintenance of Structure within Collaborative Social Media Platforms

Eva Zangerle

submitted to the Faculty of Mathematics, Computer Science and Physics of the University of Innsbruck

in partial fulfillment of the requirements for the degree of “Doktor der technischen Wissenschaften”

Advisor: Univ.-Prof. Dr. Günther Specht

Innsbruck, December 2012

Abstract

During the last decade, the web transformed from a web of information consumers to a web of information producers. In particular, the advent of online social media platforms is hugely responsible for this shift as people now actively post information in knowledge bases, engage in online communities and contribute to social media platforms. Hence, a vast amount of new information is produced each day. This publicly available data is an invaluable source of information which still is to be fully exploited. Due to the broad span of users of such systems (originating from different cultures and backgrounds, speaking different languages, etc.), the information provided features a limited amount of common structure, as e.g., objects are named differently and information is structured differently. This is a severe constraint in regards to the performance of search facilities.

This thesis proposes to facilitate recommender systems to create and maintain a common structure within collaborative social media platforms aiming at improving search performance. For this purpose, two different recommender systems for two showcase platforms are presented. The first recommender system provides recommendations for structuring information within a semistructured information system whereas the second recommender systems is a hashtag recommender system for microblogging services.

Zusammenfassung

Während des letzten Jahrzehnts hat sich das Web von einem Netz von Informationskonsumenten zu einem Netz von Informationsproduzenten gewandelt. Insbesondere die zunehmende Verbreitung von Social Media Plattformen hat großen Anteil an dieser Entwicklung. Menschen erstellen nun aktiv Beiträge in Wissensbasen, beteiligen sich in online Communities und wirken bei Social Media Plattformen mit. So werden täglich sehr große Datenmengen erzeugt, die für die Öffentlichkeit zugänglich sind. Allerdings werden diese wertvollen Datenmengen noch nicht vollständig genützt. Aufgrund der großen Diversität der Benutzer (verschiedene Kulturen, Hintergründe, verschiedene Sprachen, etc.), weisen die verfügbaren Informationen nur beschränkt eine gemeinsame Struktur auf, so sind beispielsweise gleiche Objekte oft verschieden benannt oder Information ist verschieden strukturiert. Dies wirkt sich negativ auf die Performance der Suche auf diesen Daten aus.

Diese Dissertation beschäftigt sich mit der Fragestellung, wie Recommender Systems (dt. Empfehlungssysteme) verwendet werden können, um eine gemeinsame Struktur in kollaborativen Social Media Plattformen erstellen und pflegen zu können. Das Ziel dabei ist, die Such-Performance auf diesen Daten zu verbessern. Dazu werden zwei exemplarische Empfehlungssysteme für zwei solchen Plattformen präsentiert. Das erste Empfehlungssystem stellt Empfehlung für die Strukturierung von Information in semistrukturierten Informationssystemen zur Verfügung. Das zweite Empfehlungssystem ist ein Hashtag-Empfehlungssystem für Microblogging Plattformen.

Acknowledgements

... and I hope they know I never would have made it this far on my own...

Sarah, Robert and Wolfi, thank you for making these last years the most memorable and amazing experience. Thank you for countless late night discussions, table tennis games, coffee breaks, sports, cinemas and travels. Wolfi, wow—what a rollercoaster ride these years have been. You're the one I owe the most, you're my rock. Thank you for believing in me when I didn't. Sarah, you're the best! Thank you for the kitchen table evenings, the (non-computer science) dinner conversations, everything. Robert, we've come a long way together from the first day of our studies more than a decade ago. Since this very day, you've been the most patient, generous and fun companion. Thank you!

Thank you to all the DBIS members: Dominic, Michi, Seppi, Niko, Gabi, Peter—you're also hugely responsible for making my time at DBIS enjoyable, fun and entertaining. Thank you for your help and the distractions.

A special "thank you" goes to my parents and my sister who supported me regardless of the fact that they do not fully understand what I've been working on throughout the last years.

I also want to thank my advisor Günther Specht for giving me the opportunity of writing this thesis and his support during this time.

This work was partially funded by the Tiroler Wissenschaftsfonds and the Nachwuchsförderung of the Leopold-Franzens-University Innsbruck.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt durch meine eigenhändige Unterschrift, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Stellen, die wörtlich oder inhaltlich den angegebenen Quellen entnommen wurden, sind als solche kenntlich gemacht. Die vorliegende Arbeit wurde bisher in gleicher oder ähnlicher Form noch nicht als Magister-/Master-/Diplomarbeit/Dissertation eingereicht.

Datum	Eva Zangerle

Contents

CHAPTER 1

Introduction

1.1. Motivation

Throughout the last years, the web transformed from a web where most users were simple consumers passively viewing website to a more active and participatory medium. Users are now able to actively contribute to and engage in knowledge bases, online communities and social platforms. Such community-based systems like collaborative information systems or social media platforms share the fact that huge amounts of users contribute to a common knowledge base, i.e. the online encyclopedia Wikipedia[1] was created solely by the effort of a committed community. Such communities consist of a huge number of users originating from various different countries, backgrounds and cultures. Moreover, these users may also speak different languages. The more heterogeneous the users are, the more heterogeneous the vocabulary used within online communities is as users tend to use differing and heterogeneous vocabulary. E.g., one user may use the term “population” whereas another user may describe the same fact by using the term “inhabitants”. Also, people tend to structure information differently.

[1]`http://www.wikipedia.org`

Due to such differences concerning the usage, collaborative information systems and social media platforms are not able to achieve their full potential in regards to search facilities. Searching an information system for e.g. all cities having more than 100.000 inhabitants requires knowledge about the stored information. Such a query can only lead to a correct and precise result set if the stored information is based on a homogeneous vocabulary, i.e. all users stick to the term "population". In this case, querying for "population" leads to the correct search result. However, if some users also used the synonymous term "inhabitants", these facts may not be incorporated in the search results even though the information is semantically relevant for the query. Therefore, the more users adhere to a common structure, the higher is the benefit. This thesis is concerned with the question of how the users of such collaborative information systems can be supported aiming at creating and maintaining a homogeneous structure within the system's data. In particular, recommender systems are facilitated for this task.

The notion of a *recommender system* describes a system which automatically provides its users with useful recommendations for items they might be interested in. Traditionally, recommender systems are used in online shops where clients are pointed to further products. Another scenario is the recommendation of movies for the users of a movie database. Such recommendations are computed by applying similarity measures to the stored data in order to either find items similar to those the user bought before or to find users with similar preferences to further deduce item recommendations. Throughout the last years, the research field of recommender systems has gained significant importance in both economy and academia. Recently, recommender systems have proven to be suited not only for the traditional, above mentioned use cases. Recommender systems have e.g. also been implemented for the suggestion of friends in social networks.

In this thesis, we propose that recommender systems can be facilitated for the creation of a common structure within collaboratively created data sets. Therefore, two different recommender systems are presented. The first recommender system proposed in this thesis is related to *microblogging platforms*. Such platforms enable users to share short messages with the public and gained massive popularity during the last years. On the Twitter microblogging platform[2], hundreds of millions of such microblog entries are issued each day. This massive amount of information is hardly structured and hence, users proposed to make use of so-called hashtags. Such hashtags are keywords describing the topic of the tweet and can be recognized by the leading hash-sign (`#`) as e.g., the hashtag `#www2012` is used for posts concerned with the WWW 2012 conference. Again, such hashtags may be chosen freely by the users and this

[2]`http://www.twitter.com`

naturally leads to a heterogeneous sets of hashtags. Therefore, we propose a recommender system which aims at providing users of microblogging services with suitable recommendations for hashtags for their tweets already during the composition of the according microblog post. The second recommender system is facilitated in a *semistructured information system* where information is stored as triples. E.g., the fact that Innsbruck is located in Austria can be modelled by the triple (*Innsbruck, locatedIn, Austria*). The advantage of such information systems is that they are very flexible as any kind of information can be stored as a triple and as such, features structure. However, this flexibility comes at a price as it enables users to freely choose the three components (subject, predicate and object) of such a a triple. This may lead to synonymous entries which are hard to handle in terms of search performance. Hence, the goal of our recommender system is to provide recommendations for subjects, predicates and objects aiming at encouraging the users to adhere to a common structure while at the same time not constraining them.

1.2. Aims and Research Questions

The aim of this dissertation is to facilitate recommender systems for the creation and maintenance of structure within information systems. Therefore, efficient, personalized algorithms for the recommendation of highly suitable objects are developed which aim at providing a homogeneous structure of the contained information based on already existent usage data. Hence, through the homogeneous schema, the search performance can be improved.

In particular, these aims can be characterized by the following research questions which are addressed in this thesis:

RQ1: Can hashtag recommendations be leveraged to enhance structure in microblog entries?

RQ2: Can structure within semistructured data be created and maintained efficiently by content-aware recommendation mechanisms?

1.3. Contributions and Published Work

The contributions made in the course of this thesis can be divided into two fields of research:

In the field of semistructured information systems, our approach is the first work which facilitates a recommender system for proactively preventing schema proliferation in semistructured information systems. The main contribution of our work lies in showing that recommender systems are capable of ensuring a homogeneous semistructured data set. In the course of our research, we identified different types of recommendations required for the creation of a homogeneous data set: (i) content-aware structure recommendations, (ii) content recommendations and semantic refinements and (iii) intelligent autocompletion. For each of these types, we developed suitable recommendation algorithms. These algorithms were subsequently implemented and a fully functional prototype called *SnoopyDB* was developed which incorporates these algorithms. Based on this prototype, test-user experiments were conducted for the evaluation of the proposed algorithms. As we identified content-aware structure recommendations as the most influential type of recommendations in regards to the data set created, we conducted offline evaluations in order to evaluate the performance of the recommendation algorithms based on a large data set. The results of these evaluations were already published at multiple international conferences. At the CTS conference (2011 International Conference on Collaboration Technologies and Systems), the paper “The Snoopy Concept: Fighting Heterogeneity in Semistructured and Collaborative Information Systems by using Recommendations” [35] was published and was also nominated for the best paper award. At the ACM Recommender System Conference 2010, the most widely known conference on Recommender Systems, the paper “Recommending Structure in Collaborative Semistructured Information Systems” [117] was published. This paper describes the algorithms underlying the SnoopyDB approach. Further details about these approaches have been published as a poster at the ACM Hypertext Conference 2010 [36] and at the Workshop “Grundlagen von Datenbanken 2010” organized by the Deutsche Gesellschaft für Informatik (German computer science society) [34]. An extensive survey concerned with the different approaches dealing with the process of creating a common structure within semistructured knowledge was published as a book chapter [116].

In the field of hashtag recommendations for microblogs, we published the first work concerned with the recommendation of hashtags as an aim for creating a more homogeneous set of hashtags. The main contribution of this work is the demonstration of the feasibility of recommendation mechanisms for the creation of a homogeneous set of hashtags. The brevity of microblogging entries and the fact that the recommendations are based on the content of the tweet place pose special challenges in regards to the algorithms underlying the recommender system. Therefore, we developed algorithms which are able to cope with the brevity of tweets and the according characteristics of microblogs. In the course of this thesis we identify string similarity measures which are particularly suited for the recommendation task. Furthermore, we crawled a data set containing more than 380 million tweets for the analysis of the hashtagging behavior of Twitter users and also as a basis for the recommendation process. This data was also used for the evaluation of the proposed solution. Moreover, a framework for the recommendation of hashtags for a given tweet was developed which incorporates the proposed algorithms for the recommendation of hashtags. The results of this research have been published and presented at the Workshop on Semantic Adaptive Social Web 2011 at the 19th International Conference on User Modeling, Adaptation and Personalization. The paper's title is "Recommending #-tags in Twitter" and it contains the basic algorithms for the recommendation of hashtags [118]. The optimized algorithms and a more extensive evaluation have been published in "Using Tag Recommendations to Homogenize Folksonomies in Microblogging Environments" at the Third International Conference on Social Informatics 2011 [119].

Besides these two fields, also contributions have been made in the field of main memory graph storage and music recommendation. However, these works are out of the scope of this thesis. Below all works published throughout the course of the PhD studies are listed:

- E. Zangerle, W. Gassler, and G. Specht. *Recommending Structure in Collaborative Semistructured Information Systems.* In RecSys '10: Proceedings of the third ACM Conference on Recommender Systems, pages 141–145. ACM, New York, NY, USA, 2010.

- E. Zangerle, W. Gassler, and G. Specht. *Recommending #-tags in Twitter.* In Proceedings of the Workshop on Semantic Adaptive Social Web 2011 in connection with the 19th International Conference on User Modeling, Adaptation and Personalization, UMAP 2011, pages 67–78. CEUR-WS.org, ISSN 1613-0073, Vol. 730, available online at http://ceur-ws.org/Vol-730/paper7.pdf, urn:nbn:de:0074-730-4, 2011.

- E. Zangerle, W. Gassler, and G. Specht. *Using Tag Recommendations to Homogenize Folksonomies in Microblogging Environments.* In Social Informatics, volume 6984 of Lecture Notes in Computer Science, pages 113–126. Springer, Berlin, Heidelberg, New York, 2011.

- E. Zangerle, W. Gassler, and G. Specht. *Exploiting Twitter's Collective Knowledge for Music Recommendations.* In Proceedings of the 2nd Workshop on Making Sense of Microposts (#MSM2012): Big Things come in Small Packages in connection with the 21st International Conference on World Wide Web, 2012.

- E. Zangerle and W. Gassler. *Dealing with Structure Heterogeneity in Semantic Collaborative Environments.* In S. Brüggemann and C. d'Amato, editors, Collaboration and the Semantic Web: Social Networks, Knowledge Networks and Knowledge Resources, pages 1–20. IGI Publishers, Hershey, Pennsylvania (USA), 2012.

- R. Binna, W. Gassler, E. Zangerle, D. Pacher, and G. Specht. *SpiderStore: Exploiting Main Memory for Efficient RDF Graph Representation and Fast Querying.* In Proceedings of the 1st International Workshop on Semantic Data Management (SemData) in connection with the 36th International Conference on Very Large Data Bases (VLDB 2010). CEUR-WS.org, ISSN 1613-0073, Vol. 637, available online at http://ceur-ws.org/Vol-637/paper3.pdf, urn:nbn:de:0074-637-5, 2010.

- R. Binna, W. Gassler, E. Zangerle, D. Pacher, and G. Specht. *SpiderStore: A Native Main Memory Approach for Graph Storage.* In Proceedings of the 23nd Workshop Grundlagen von Datenbanken. CEUR-WS.org, ISSN 1613-0073, Vol. 733, available online at http://ceur-ws.org/Vol-733/paper_binna.pdf, urn:nbn:de:0074-733-4, 2011.

- W. Gassler and E. Zangerle. *Recommendation-Based Evolvement of Dynamic Schemata in Semistructured Information Systems.* In Proceedings of the 22nd Workshop Grundlagen von Datenbanken. CEUR-WS.org, ISSN 1613-0073, Vol. 581, available online at http://ceur-ws.org/Vol-581/gvd2010 3 3.pdf, urn:nbn:de:0074-581-7, 2010.

- W. Gassler, E. Zangerle, M. Bürgler, and G. Specht. *Snoopytagging: Recommending Contextualized Tags to Increase the Quality and Quantity of Meta-Information.* In Proceedings of the 21st International Conference Companion on World Wide Web. ACM, New York, NY, USA, 2012.

- W. Gassler, E. Zangerle, and G. Specht, editors. *Proceedings of the 23rd GI-Workshop on Grundlagen von Datenbanken.* CEUR-WS.org, ISSN 1613-0073, Vol. 733, available online at http://ceur-ws.org/Vol-733/, urn:nbn:de:0074-733-4, 2011.

- W. Gassler, E. Zangerle, and G. Specht. *The Snoopy Concept: Fighting Heterogeneity in Semistructured and Collaborative Information Systems by using Recommendations.* In Proceedings of the 2011 International Conference on Collaboration Technologies and Systems (CTS 2011), pages 61–68. IEEE Computer Society, Piscataway, NJ, USA, 2011.

- W. Gassler, E. Zangerle, M. Tschuggnall, and G. Specht. *SnoopyDB: Narrowing the Gap between Structured and Unstructured Information* *using Recommendations.* In HT'10, Proceedings of the 21st ACM Conference on Hypertext and Hypermedia, pages 271–272, 2010.

- A. Larcher, E. Zangerle, W. Gassler, and G. Specht. *Key Recommendations for Infoboxes in Wikipedia.* Website of the 22nd ACM Conference on Hypertext and Hypermedia. Poster Presentation, available online at http://www.ht2011.org, 2011.

1.4. Methodology

The research presented in this thesis was conducted following a research methodology which is split into four steps and is applied to each of the research questions:

- **Literature Review:** The first step is a thorough review of literature related to the according research question. For the field of semistructured information systems, we review work that was published in the fields of collaborative information systems, collaboratively created data sets, structure extraction and tag recommendations. As for the microblogging environment, we review literature related to microblogging, hashtags, recommender systems based on microblogging and social and behavioural phenomena in the context of microblogging platforms. The review of publications related with tag recommendations (already conducted for the semistructured information system research) is also applicable for microblogging data.

- **Development of Recommendation Algorithms:** In this step, suitable recommendation algorithms are designed. The findings of the literature review are transferred to the context of microblogs and semistructured data and adapted to the specific needs of these two fields. The algorithms are evaluated based on prototypical implementations and the findings of the evaluations are used to iteratively improve the algorithms and hence, improve the quality of the recommendations.

- **Implementation of a Prototype:** For each of the proposed approaches, a proof-of-concept implementation was done in order to be able to (i) proof the suitability of the proposed recommendation algorithms and (ii) allow for an iterative improvement of the algorithms.

- **Evaluation of the Approach:** Based on the previously described prototypes, each approach was thoroughly evaluated by either user-tests (online evaluation) or by automatically computing evaluation metrics (offline evaluation). The findings of these evaluations are directly used for a further optimization of the designed algorithms and hence, are also part of the iterative development cycle.

These steps are conducted for both of the research questions, the research within both of these fields (microblogging and semistructred information systems) contributed to each another. E.g., the co-occurrence approach facilitated for the semistructured recommender systems was partly also used for the recommendation of hashtags for microblogging platforms and the ranking of the according recommendations. Furthermore, knowledge gained during the course of designing the algorithms for semistructured data was transferred to the algorithms for the microblogging environment.

1.5. Thesis Outline

This dissertation is structured as follows. Chapter 2 features an introduction to recommender systems and describes the aims of such systems and subsequently the different approaches for the recommendation of items. In Chapter 3, we present how Recommender Systems can be used for microblogging services in regards of providing suitable recommendations for hashtags aiming at creating a more homogeneous hashtag vocabulary. Chapter 4 is concerned with the facilitation of recommender systems for information systems based on the semistructured storage paradigm. In particular, this chapter describes how recommender systems can be facilitated for the creation and maintenance of a common and uniform structure within a multi-user semistructured information system. Chapter 5 concludes this thesis and discusses future work.

CHAPTER 2

Recommender Systems

Generally, *recommender systems* are concerned with recommending certain suitable items to the users of the system [87] and have been a popular research topic for roughly the last two decades. Due to the uprising of the web, social media platforms and the increasingly collaborative nature of the web, recommender systems have gained tremendous popularity in both academia and industry.

The first research project proposing a recommender system was facilitated within the Tapestry system [38]. This system was an alternative email system which provided facilities to filter (and search) documents attached to these emails based on annotations of other users who have already studied and subsequently annotated these documents. The authors called the approach underlying the Tapestry system "Collaborative Filtering" as the filtering of documents is based on the collaborative effort of annotating the documents accomplished by all participating users. Since these first steps in this field, recommender systems have evolved into one of the most dominant paradigms on the web as the range of applications of recommender systems spans from the recommendation of products to the recommendation of music tracks, friends on social network platforms or even people on online dating services. Recommender systems are of high value for industry and commerce. Thus, many recommender systems evolved in industrial settings. On the Amazon website[1] users are provided with recommendations for books or other products they might be interested in based on their previous purchases, products they showed interest in or based on the profiles of other similar users [68]. Amazon features a huge recommender system in terms of both recommendable items and the number of users of the system.

A second mayor application of recommender system which evolved especially during the last few years is facilitating recommender systems as a substitute for traditional, keyword-based web search in terms of an information filter for online data. Recommender systems hence contribute to coping with the information overload problem [88]. The customized recommendation of news [57, 84] is a popular example for such a recommender system as the sheer amount of news available on the web is not easily searched for news articles of interest for a particular user. In such a system, news items are filtered based on the user's preferences and her social profile.

In the following section, recommender systems and their aims are firstly formally defined and secondly, the different types of recommender systems are presented.

2.1. Aim of Recommender Systems

The basic aim of a recommender system is to provide its users with a set of personalized, suitable recommendations for items. As input for such a recommender system, three main sources can be exploited: (i) the set of users of the system and their profile describing their preferences and characteristics, (ii) the set of items available for recommendation and their respective features and metadata and (iii) the set of user-item relations deduced from the two previous sources. In order to assess a user's preferences, either explicit or implicit feedback of this particular user may be used to create a user profile describing her preferences. *Explicit feedback* is gathered from user-ratings for items, like the 5-star rating system for books and other items on Amazon which are a very good indicator for whether a certain user liked a certain item or not. *Implicit feedback* relies on more subtle, mostly behavioral information as the user does not have to enter feedback explicitly. The fact that a user e.g. purchased or bookmarked a certain item or simply clicked the description of a certain item can be collected and used as implicit feedback.

[1]`http://www.amazon.com`

A recommendation task can formally be defined as follows[2] [3]:

Let $\mathcal{U}$ be a set of all users of a system and let $\mathcal{I}$ be the set of all items within the system (each of these items may be recommended). The utility function $s(u, i)$ can then be used to estimate how useful and suitable a certain item $i \in \mathcal{I}$ might be for a certain user $u \in \mathcal{U}$. The function is defined as $s : \mathcal{U} \times \mathcal{I} \to \mathcal{R}$ where $\mathcal{R}$ is a non-negative integer or real number (mostly within a given range) representing a utility value. It is important to note that $s(u, i)$ is not available for each and every pair (u, i) (e.g., due to information sparsity in the case of a new user who has not specified any preferences yet). Therefore, only a subset of the $\mathcal{U} \times \mathcal{I}$ space is specified. The missing utility values for items the user has not actively or passively rated yet have to be predicted.

Based on these definitions, Mobasher [74] specifies the *profile* of a certain user $u \in \mathcal{U}$ as an n-dimensional vector of ordered pairs (n being the number of items in $\mathcal{I}$), where the utility function s assigns a utility value to items $i \in \mathcal{I}$ for each user u (see Equation 2.1).

$$u^{(n)} = \langle (i_1, s(u, i_1)), (i_2, s(u, i_2)), ..., (i_n, s(u, i_n)) \rangle \tag{2.1}$$

In the case of a system with explicit ratings, the function $s(u, i)$ can be seen as a rating function, according to Mobasher. I.e. all items the user has actively rated feature the item's rating as the utility value for the corresponding item. Thus, the user is characterized by her preferences.

Mobasher describes the set of all user profiles as $\mathcal{UP}$, the set of n-dimensional user profile vectors (which may be empty in the launch phase of the system). The task of a recommender system can then be defined as a mapping of users $\mathcal{U}$ to a set of recommended items $\mathcal{P}(\mathcal{I})$ which are computed based on a subset of other user profiles $\mathcal{P}(\mathcal{UP})$. The recommendation function REC is thus of the following form:

$$REC : \mathcal{P}(\mathcal{UP}) \times \mathcal{U} \to \mathcal{P}(\mathcal{I}) \tag{2.2}$$

The set of all user profiles forms a user-item matrix $[s(u_k, i_j)]_{m \times n}$, where the entries of the $m \times n$ matrix are the utility values for the respective items, as can be seen in Example 2.1. This matrix shows a user-item matrix for explicit user feedback in a 0 to 5 stars rating system. In contrast, a user-item matrix originating from implicit feedback which features only boolean values (e.g., for a user having visited a certain product description page or not) can be seen

[2]This definition is aimed at collaborative filtering tasks. However, the tasks of other recommender systems can be defined analogously.

$$\begin{array}{c} \\ u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{m-1} \\ u_m \end{array} \begin{array}{c} \begin{array}{cccccc} i_1 & i_2 & \dots & \dots & i_{n-1} & i_n \end{array} \\ \left(\begin{array}{cccccc} 1 & & & 4 & & \\ & & 2 & 3 & & \\ & 3 & & & & 2 \\ 0 & & & 5 & & \\ 0 & & & 5 & & 1 \\ & & & 2 & 0 & \end{array} \right) \end{array}$$

Example 2.1.: User-item Matrix for Explicit User Feedback

$$\begin{array}{c} \\ u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{m-1} \\ u_m \end{array} \begin{array}{c} \begin{array}{cccccc} i_1 & i_2 & \dots & \dots & i_{n-1} & i_n \end{array} \\ \left(\begin{array}{cccccc} 1 & & & 1 & & \\ & & 1 & 1 & & \\ & 1 & & & & 1 \\ 1 & & & 1 & & \\ 1 & & & 1 & & 1 \\ & & & 1 & 1 & \end{array} \right) \end{array}$$

Example 2.2.: User-item Matrix for Implicit User Feedback

in Equation 2.2[3]. As can be seen, such matrices are typically very sparse as it is hardly possible for a user on Amazon to visit or rate all items available.

Based on these definitions, the recommendation task for a certain user u_k can be formalized as shown in Equation 2.3, where *up* is a subset of user profiles relevant for the recommendation task. From this equation it becomes clear that the task is to find items which this particular user has not previously rated aiming at maximizing the utility value for these items according to some utility function s. The function *arg max* is used to determine the maximum utility value for the given items.

$$REC(up, u_k) = \{\ i \mid s(u_k, i) = arg\ \ max_{i \in \mathcal{I}}\ \ s(u_k, i)\} \tag{2.3}$$

Hence, the recommendation task aims at finding items which reach a maximum utility value, i.e. to find the most suitable and useful items for a certain user which have not been rated yet by the according user. Mostly, the top-x most useful items are finally recommended to the user.

[3]It is important to note that implicit feedback may also feature non-boolean values, e.g. when using the amount of time a certain user spent on a certain product's page as implicit feedback.

2.2. Overview

In order to fulfill the previously described recommendation task, various approaches for the recommendation of items have been developed throughout the last two decades. The algorithms and approaches underlying typical recommender systems are classified into the following categories (as in [17, 88]):

- Collaborative filtering
- Content-based recommender systems
- Demographic recommender systems
- Knowledge-based recommender systems
- Hybrid recommender systems

Adomavicius *et al.* [3] propose three categories of recommender systems, namely collaborative filtering, content-based and hybrid approaches. Jannach *et al.* [51] omit the demographic approach. However, in order to provide a complete overview about the different recommendation methods, we distinguish between five different recommendation approaches as listed above. In the following sections, these different approaches aiming at recommending items to a specific user are introduced. As collaborative filtering is by far the most popular approach, it is discussed in more detail.

2.3. Collaborative Filtering

Collaborative filtering (CF) aims at recommending items to a certain user based on her past actions (purchase of a certain product, consumption of certain music tracks, explicit rating of certain items, etc.) and past actions of other, similar users. The term collaborative filtering was first introduced in 1992 by Goldberg *et al.* [38] for Tapestry, which was used to collaboratively filter attachments of corporate emails.

Shardanand and Maes [98] state that a collaborative filtering system is an automation of the word-of-mouth principle. This principle basically describes that recommendations are computed based on items which were rated by other users who have shown similar preferences. Hence, CF techniques aim at recommending items based on the user profiles of other users as users having shown similar interests in the past may be a good source for recommendations. CF computation is based on a matrix consisting of all users, items and the users' ratings for items. Theses matrices are referred to as user-item matrices in the course of this thesis. Such a matrix may contain either explicit or implicit rating information. Examples for user-item matrices have been shown in Example 2.1 resp. Example 2.2.

In principle, two approaches for collaborative filtering can be distinguished [3]: memory-based and model-based approaches. Both of these approaches are described in the next sections.

2.3.1. Memory-Based CF

Memory-based approaches for CF make use of the entire user-item matrix and recommendations are computed directly based on the information available from this matrix. In general, two types of recommendation tasks can be computed: user-based filtering and item-based filtering. *User-based filtering* aims at matching the current user to other users in the matrix, whose preferences are used to make predictions about the possible preferences of the current user. By extracting all new items from the most similar users, a set of recommendations is created which is then ranked and presented to the user. [3] defines the task of memory-based methods as predicting the rating for a certain item by aggregating the ratings of the top-k most similar users within the system (the k-nearest neighbours, kNN). Mostly, a weighted aggregation function (e.g., the average of all ratings weighted by the similarity of the two users) is applied for the computation of the rating predictions. Subsequently, the items with the highest predicted ratings are recommended. As for the similarity of users, [105] and [14] state that mostly either a correlation-based similarity (e.g., Pearson correlation coefficient) or a cosine-similarity measure based on user-profile vectors is applied. As for *item-based filtering*, the goal is to find the most similar items based on the user profiles of the users who rated these items [94]. The most similar items are ranked and subsequently provided to the user. Two items are similar if the same users have rated these items similarly or implicitly showed interest in these two items. Again, the similarities are computed by either Pearson correlation coefficient or the cosine similarity of the according vectors.

2.3.2. Model-Based CF

In contrast to memory-based CF, model-based CF aims at learning a predicting model for a certain user based on a user-item matrix. This learning of a model based on training data is done offline. At the time of computation of recommendations, the precomputed model is applied. As for the models underlying the computation, Breese *et al.* [14] propose two different probabilistic models: cluster models and Bayesian networks. The cluster model aims at computing a probability value assessing how likely it is that a user belongs to a certain cluster or class of users. Based on this probabilistic attribution to a certain class (featuring certain common preferences), recommendations are computed. The Bayesian network approach models the CF problem as a Bayesian network where each node represents an item and the according preferences. Based on this network, a Bayesian network is learned in order to be able to predict preferences for certain items.

Memory-based CF is a computationally very intensive task as this approach computes recommendations based on the whole matrix. The scalability of such approaches is somewhat limited as the size of the underlying matrix grows with the number of users and items within the system. In contrast, model-based CF are better able to cope with scalability issues as models are trained once and then, these models can be applied. However, these models require a training phase ex ante which is not the case for memory-based CF.

The main problem with which both CF approaches have to cope with is sparsity of data. As most users traditionally only have rated a very small fraction of all available items, the user-item matrix is traditionally immensely sparse. Especially during the initial phase of a system where hardly any user ratings are available, the quality of the recommendations due to the lack of available information for the computation of recommendations is not satisfying. This is also the case for new items which have not been rated at all and users who have not (implicitly or explicitly) stated any preferences yet. This problem is also referred to as the *cold-start problem* [51].

2.4. Content-based Recommender Systems

Content-based recommender systems [3, 27, 51, 82, 88] are focused on recommending similar items. A content-based recommender system computes recommendations by finding items which are similar to the item preferences of the user (traditionally the preferences are modelled in a user profile). Schafer *et al.* call this *item-to-item correlation* [95]. As for the profile of the user, the items a user previously liked or rated are used to build a profile for the user which represents her interests and preferences. The actual recommendations are based solely on the features of the items and the according similarity of items (in contrast to item-based collaborative filtering where user profiles are used to determine the similarity of items). For a book recommender system, such features may be the genre, the author or the topic of the book. For the computation of similarity, different approaches have been facilitated in the past. A very simple approach is to use the keywords of the features associated to certain items and compute a set-based similarity coefficient of the sets (e.g., the Dice coefficient) of keywords for items in order to compute the similarity of two items. A popular approach for content-based recommender system is modelling the features of two items as vectors where the components of the vector represent the item's features (the components of the components may also be weighted in order to represent the importance of certain features). The similarity of two items is computed by determining the cosine of the angle between these two feature vectors. One example is the Fab system [9], which aims at recommending websites to users. It uses the most important words representing the according websites as feature vectors which are used to compute the similarity of the according websites.

The main advantage of content-based recommender systems is that these approaches do not have to cope with the cold-start problem as the features of the recommendable items are well-known. However, ratings and preferences of other users within the system form a valuable source of relevant information which is not exploited at all in content-based recommender systems.

2.5. Demographic Recommender Systems

Demographic recommender systems make use of demographic data about its users, like age, sex, marital status etc. Such demographic information can be exploited to create classes of users for which recommendations can be computed. The work by Pazzani [83] describes a framework which aims at recommending websites to users. Besides both content- and collaborative recommendation approaches, also a demographic recommender system is presented in this work. This recommender system attempts to extract demographic features (in this case age, sex and area code) from the user's websites and leverage this information for creating relations between items (websites) and classes of users, which are formed based on their demographic features. Based on these relations, recommendations are computed. However, demographic recommender systems have not been very popular in research as demographic classes only can provide a rough personalization of recommendations.

2.6. Knowledge-based Recommender Systems

Knowledge-based recommender systems [17, 51, 88] are focused on complex user needs and how to find items matching these user needs. The more complex a user need is, the less likely it is to find other like-minded users in order to compute recommendations based on their previous preferences and actions. Therefore, knowledge-based recommender systems do not rely on other user's experiences and actions, they are rather based on a reasoning and inference task based on constraints, patterns and rules. Also, functional knowledge in regards to how certain items meet the user's requirements is used. Rules and patterns are defined in advance by a domain expert and subsequently exploited for recommendations [95]. Knowledge-based recommender systems are mostly directly related to conversationally eliciting the user's needs and preferences. This allows for narrowing down the number of items corresponding to the user's preferences similar to facet-based search. This can also be achieved by actively asking the user for his preferences in regards to certain features of the desired item. Consider a car recommender system. People do not buy cars very often and hence, information about the need of a certain user is sparse and thus, interaction to elicit her preferences is required. Based on this gained knowledge, rules, patterns and constraints are applied in order to compute suitable recommendations.

Knowledge-based recommender systems are mostly used to complement the shortcomings of another type of a recommender system. E.g., for a content-based recommender system or CF a knowledge-based recommender system may help to deal with the cold-start problem as no previous user actions have to be present in the system in order to be able to compute recommendations. However, the creation and formulation of the rules, patterns and constraints underlying such a recommender system is very expensive and the knowledge of a domain expert is required. Furthermore, such systems are rather inflexible and static as changes in regards to recommendable items require a previous (re)definition of rules, patterns and constraints.

2.7. Hybrid Recommender Systems

Based on the recommender system approaches introduced in Sections 2.3–2.5, hybrid recommender systems facilitate various different recommender approaches and combine these to one single, hybrid approach. The main goal of such a combination is to exploit the advantages of the different approaches while at the same time avoiding the disadvantages of a certain approach by the advantages of another. If e.g., no profile information about a certain user is present in a recommender system based on collaborative filtering, demographic approaches can be used to provide a basic profile for the user until the user provided the required information by e.g. rating certain items. In [100], a hybrid approach of collaborative filtering and content-based filtering is used to provide personalized hypermedia content (interlinked text, images, audios or videos) to the users of the system. In this system, the utility value of items is defined by the intersection of the Gaussian curves presenting the user's preferences and the characteristics of a certain item. An extensive survey about hybrid recommender systems is provided by Burke [17].

2.8. Conclusion

In this section we introduced the research area of recommender systems. Therefore, we first formalized the task of a recommender systems. Subsequently, the state-of-the-art approaches for the computation of recommendations were discussed.

CHAPTER 3

Microblogs

Microblogging services have become immensely important throughout the last years as they allow users to easily share their thoughts to the public. In general, microblogs are a special form of blogs as microblog messages are shorter than traditional blog posts and mostly feature messages of a maximum of 140 characters. These blog messages are subsequently available for the public. The most successful and popular microblogging platform is Twitter[1] which currently serves more than 140 million active users who publish about 340 million[2] posts each day. The motivation of users to participate in such platforms are manifold [53]. People make use of such services to stay in touch with friends, to follow news and latest events, to gather information or to simply tell the world what they are up to. Hence, Twitter has also become an important social network. As of March 2012, Twitter currently is the third most successful social network in regards to total visits in the U.S.[3]. Due to the high volume of data published each day, microblogging platforms also have

[1]http://www.twitter.com

[2]https://business.twitter.com/en/basics/what-is-twitter/

[3]http://www.dreamgrow.com/top-10-social-networking-sites-by-market-share-of-visits-march-2012/

become an important source of information. E.g., when dealing with a certain problem, the Twitter community is often used for finding solutions by simply searching for answers or posting questions to the community. Furthermore, Twitter is an important news medium where news are spread at a very high pace [61].

Despite the huge volume of tweets posted, this data hardly features structure in terms of categorization of tweets. The only structural information available are so-called hashtags which are a means to add simple keywords as a part of the tweet. However, as hashtags may be chosen freely by the users, the hashtag vocabulary is heterogeneous. Searches for hashtags in order to find tweets concerning a certain topic may result in a search result featuring low recall due to this heterogeneity of hashtags. Therefore, our work aims at providing users with recommendations for hashtags and therefore add structure to microblog entries aiming at a more homogeneous set of hashtags enabling better search performance.

In this chapter, we present our approach for the creation of structure facilitated by recommendations in microblogging environments. Therefore, we introduce the most popular microblogging platform Twitter which is used as a showcase example for our approach in Section 1. In Section 2, we analyse the data sets underlying both the recommendation algorithm and also the evaluation thereof. Section 3 features information about hashtags, their usage and their characteristics. Subsequently we present the proposed algorithms for the facilitation of recommendations for the creation of structure in Section 4. We present and elaborate on the evaluation of the proposed approach in Section 5. Section 6 contains a description of the architecture of the recommender system. Important related work and related approaches are described in Section 7. Section 8 features future work and concludes this chapter.

3.1. Background: Twitter

The following section introduces the most popular microblogging platform Twitter and its features. Besides Twitter, also other microblogging services like Jaiku, Pownce and Google Buzz were founded. However, as of January 2012, all of these services have been shut down. Identi.ca is a microblogging service which is very similar to Twitter, however it is Open Source and far less widely used as Twitter. Lately, also enterprise microblogging systems are increasingly deployed [89].

Generally, Twitter was founded in 2006 aiming at providing a microblogging service for messages of a maximum length of 140 characters. Within the last years, Twitter gained massive popularity within social networking services and reached up to 140 million active users[4]. The number of users steadily increases and so does the number of tweets sent per day. As of 2011, Twitter was faced with usage peaks of more than 8,000 tweets per second[5] and a total amount of one billion tweets per three days[6].

As for the motivations of users to actively participate in the Twitter network, Java *et al.* [53] identified the following intentions of users:

- *Daily chatter*: The Twitter platform serves as a means to keep in touch with friends and keep them updated. According to Java *et al.*, this is the main motivation for users to participate in the Twitter network.
- *Conversations*: Twitter is also used to communicate with certain users in direct conversations via the mention-functionality (see below for a description of this functionality).
- *Information sharing*: Users also use Twitter to propagate information (mostly in the form of posted URLs).
- *News reporting*: Twitter is also facilitated to propagate news or to comment on news items.

Similarly, Naaman *et al.* [76] found that Twitter is mostly used for information sharing, posting opinions (or complaints), statements and random thoughts and—being to most dominant content category of the sampled tweets—“me now” tweets, which basically report on the status of the user (e.g., if the user tweets that she is tired or on her way home). Interestingly, the authors found that in the case of mobile-posted tweets, 51% are “me now” tweets whereas when the tweet is posted non-mobile,s 37% are “me now” tweets. Based on these findings, the tweeters are clustered into two clusters: so-called informers and meformers. Users of the former cluster (20% of all users) aim at spreading information whereas users of second cluster (80% of all users) mostly tweet about themselves and are more self-centered.

The Twitter microblogging platform provides a variety of functions to its users. These functionalities and characteristics are briefly explained in the following section.

[4]`https://business.twitter.com/en/basics/what-is-twitter/`

[5]`http://yearinreview.twitter.com/de/tps.html`

[6]`http://blog.twitter.com/2012/03/twitter-turns-six.html`

Tweet

A tweet generally is a short message which is posted on the Twitter platform. The maximum length of such a message is 140 characters. The tweet is then published on the user's personal timeline (a chronological list of the most recent tweets of the user and his followees (see below)) and automatically broadcasted to all of the user's followers (see below for a description of the follower-followee-principle). Below an exemplary tweet is shown in which Barack Obama thanks all his voters for being re-elected.

> `This happened because of you. Thank you.`
>
> posted by @BarackObama on 2012/11/07

Follower

The concept of followers is a crucial functionality on the Twitter platform. It allows users to "follow" other users which means that if user A follows user B, all tweets of user B are shown in the timeline of user A. In this relationship, user A is the follower and user B is the followee. Such a follower-followee connection is unidirectional which means that if user A follows user B, this does not necessarily imply that also user B follows user A. Moreover, following a certain user does not require permission to do so. As users can freely decide which users to follow, they can follow other users based on their interests and thus, receive all the tweets of users she is interested in.

According to the findings of Weng *et al.* in [113], the motivation of users to follow a certain user is a shared topic interest between the follower and the followee. Additionally, the authors state that a followee following back a follower is also based on the fact that both users share the same interests. The authors showed that there is a homophily between followers and followees. The concept of homophily basically implies that people tend to be part of social networks in which the other members are similar to the user, e.g. in regards to attitudes, behaviour, interests, education, sex, gender or race [71]. According to Weng *et al.*, 72.4% of all users follow back more than 80% of all of their followers (evaluated on a data set of 1,021,039 tweets posted by the top-1000 Singapore-based Twitter users). More interestingly, Kwak *et al.* [60] found that 77.9% of all relations on Twitter are unidirectional (in a data set of 1.47 billion follow relations of 2010). Additionally, Java *et al.* [53] identified three categories of Twitter users in regards to their following-behaviour: information sources, friends and information seekers. Information sources are users who act as hubs and have a large number of followers (typically significantly more followers than followees) and tweet regularly. Friends are users who have a relatively similar number of followers and followees as they mainly are connected in reciprocal relationships. Information seekers are users who follow a large number of users, however only tweet rarely. A very similar categorization has also been observed in [59], where information sources are named broadcasters and friends are called acquaintances. However, as a third category, the authors add miscreants (evangelists) which are users who follow and contact a very high number of users aiming at getting attention and new followers.

User's Timeline

A user's timeline is a list of all tweets of a user and the users she follows. The timeline is a chronological stream of all these tweets and is aimed at giving users an immediate overview. A screenshot of the timeline of the user `@evaz23` is shown in Figure 3.1.

Replies and Mentions

A reply message is a tweet which is directed to another specific user. This is realized by starting the tweet with the recipient's username which is preceded by a @-sign. This way, (public) directed conversations can be facilitated. Within Twitter messages, users can similarly be mentioned by simply stating their user name preceded by a @-sign. Hence, mentioning users is equal to replying to a certain tweet. In the example below, `@bellogin` sends a direct reply to the user `@tototingle`.

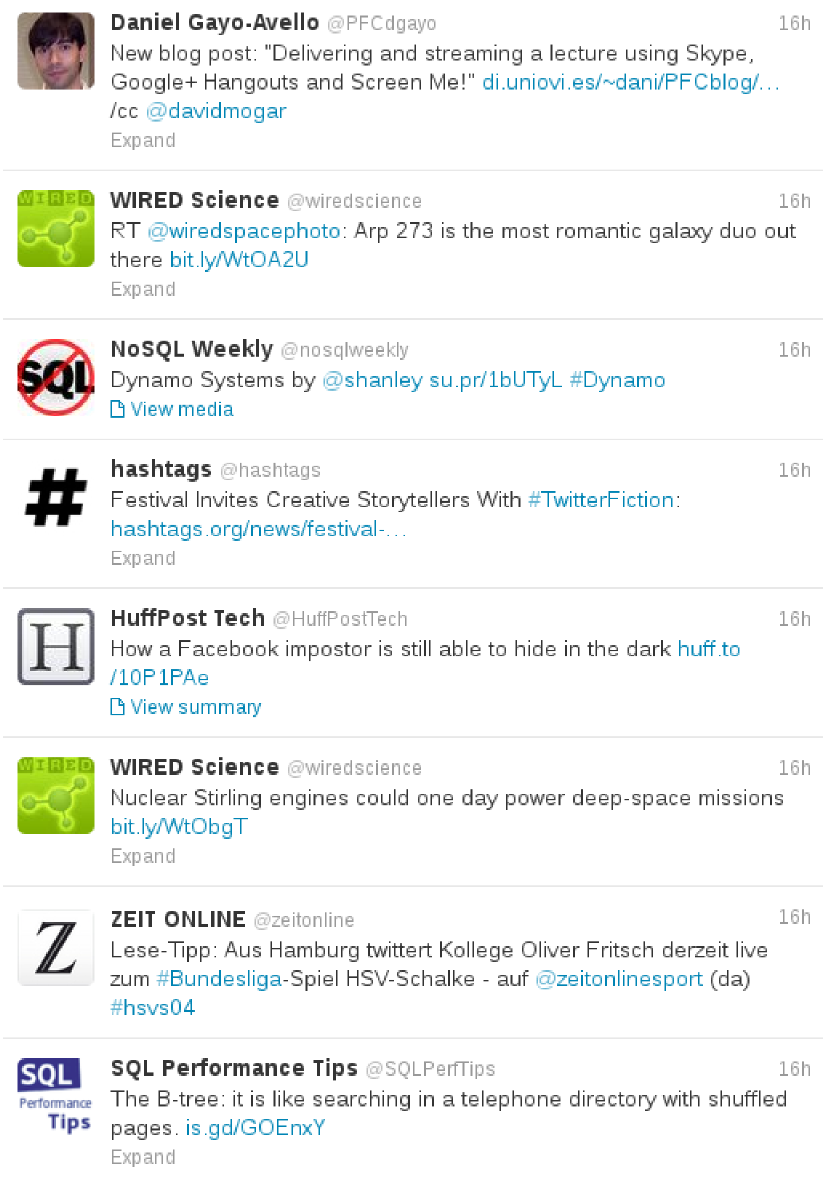
`@tototingle here you have it,it's a comparison of evaluation methodologies, using precision & ndcg: ir.ii.uam.es/~alejandro/201`

posted by @abellogin on 2011/09/21

Honeycutt and Herring [46] showed that the @-notation is primarily used for addressivity, i.e. to point messages to a certain user and hence, for conversational tweets. In the studies Honeycutt and Herring conducted, 90.96% of all @-usages were used for addressivity purposes. Only 5.43% of all @-signs were used for referencing and mentioning a certain user who is not necessarily taking part in the conversation. Further usages of @-signs include the specification of locations (e.g., "I'm @home") or as a part of emoticons.

Retweet (RT)

The retweet functionality enables users to simply (re-)broadcast a tweet of another user. Such a message is called a retweet and acts like a forward of a tweet to all followers of the retweeting user. Basically, the original messages


Daniel Gayo-Avello @PFCdgayo 16h
New blog post: "Delivering and streaming a lecture using Skype, Google+ Hangouts and Screen Me!" di.uniovi.es/~dani/PFCblog/... /cc @davidmogar
Expand

WIRED Science @wiredscience 16h
RT @wiredspacephoto: Arp 273 is the most romantic galaxy duo out there bit.ly/WtOA2U
Expand

NoSQL Weekly @nosqlweekly 16h
Dynamo Systems by @shanley su.pr/1bUTyL #Dynamo
View media

hashtags @hashtags 16h
Festival Invites Creative Storytellers With #TwitterFiction: hashtags.org/news/festival-...
Expand

HuffPost Tech @HuffPostTech 16h
How a Facebook impostor is still able to hide in the dark huff.to /10P1PAe
View summary

WIRED Science @wiredscience 16h
Nuclear Stirling engines could one day power deep-space missions bit.ly/WtObgT
Expand

ZEIT ONLINE @zeitonline 16h
Lese-Tipp: Aus Hamburg twittert Kollege Oliver Fritsch derzeit live zum #Bundesliga-Spiel HSV-Schalke - auf @zeitonlinesport (da) #hsvs04

SQL Performance Tips @SQLPerfTips 16h
The B-tree: it is like searching in a telephone directory with shuffled pages. is.gd/GOEnxY
Expand

Figure 3.1.: A User's Timeline

is copied and sent again. Retweeted messages are responsible for the vast speed in which messages and news are spread on the Twitter network [61]. Such retweets can be recognized by the prefix "RT" (also tweets featuring the keywords rt, retweeted, via, etc. are recognized as retweets) followed by the username of the original author of the tweet, like `RT @gatesfoundation` in the following example retweet where `@timoreilly` retweets the tweet originally posted by `@gatesfoundation`.

```
RT @gatesfoundation: Learn. Teach. Share. YouTube
Teachers--A new tool to help #teachers in the
classroom: http://t.co/Wj8udZYZ #edtech
```

retweeted by @timoreilly on 2011/09/22

Boyd *et al.* inspected the retweeting behaviour of users in [13]. They state that users make use of retweets as a form of both information diffusion and also the participation in a diffuse conversation. As such, users retweet in order to engage with others about certain topics. The authors found that retweeting users fall into two categories: preservers and adapters. Preservers are users who retweet messages without editing the original message whereas adapters edit a message before retweeting it. The most important finding of this paper is the motivation of users behind retweeting certain tweets, which are manifold:

- Users make use of retweets to inform an audience about a certain topic.
- Retweets are also used to acknowledge a certain tweet or to send a sign of agreement in response to a certain tweet.
- Retweets may also be performed as an action of loyalty or friendship.
- Another—more self-centered reason—for a certain user to retweet a certain tweet is to make herself more visible to the audience.

Metaxas and Mustafaraj [75] analysed political tweets and found that Twitter users are more likely to retweet a tweet from a politically like-minded use. In particular, in their data set 95% of all retweets of tweets originally tweeted by liberal were sent by liberal users. 99.10% of all tweets originally posted by conservatives were retweeted by conservative users.

Boyd *et al.* [13] also found that due to the constraint that a tweet can only hold 140 characters, different patterns of shortening a retweeted message in order to adhere to the maximum length of a tweet, are facilitated. Many users simply delete certain characters. The disemvoweling (vowels are removed from messages) of tweets is also very popular. However, the main method to shorten tweets is the removal of whole words.

Private Message

Besides the public timeline containing all public tweets, Twitter also allows for sending private messages. Such messages can be sent to users who are followers or followees of the user and are not accessible to other users.

Hashtags

Hashtags are a way for users to categorize their tweets as they allow for a specification of keywords, topics and categories for a given tweet. Hashtags are preceded by a hash sign (#) followed by an arbitrary combination of characters, e.g., the hashtag `#egypt` was used for tweets related to the uprise and protests in Egypt in early 2011. Another example is the hashtag used for tweets concerning the CIKM conference 2011[7] is `#cikm2011`. All tweets containing this hashtag are supposedly concerned with this conference. Therefore, when using the hashtag as a search term, tweets about this conference (in particular, those tweets which have been provided with the according hashtag), can be detected and related conversations can be followed. Such a search can be seen as an additional, dynamic timeline. In addition to the provided search capabilities, hashtags are a simple means for the creation of loosely connected communities. This means that hashtags cannot only be used passively as a consumer of tweets concerning e.g., the CIKM conference, but also can be used actively to participate in discussions evolving around the conference by specifying the CIKM hashtag in tweets concerning the conference. This way, ad-hoc communities are created implicitly as the group or community does not have to be created or formed manually it rather is a loosely coupled set of Twitter users. They consume and take part in a stream of conversation concerned with a certain topic which is solely based on the fact that all these users incorporate the same hashtag in their tweets. Still, all tweets containing a certain hashtag posted by a certain user are broadcasted to all of her followers.

Hashtags may be located at any position within the tweet. Generally, the Twitter community established two ways of using hashtags as a means for categorizing a tweet:

- As a part of the text where one or more words of the actual tweet content are simply preceded by a hash sign. E.g., in the following tweet, the hashtag `#cikm2011` is also part of the sentence and hence, less space is used as opposed to additionally specifying hashtags at the end of the tweet.

```
Only two weeks to go before the end of the #cikm2011
regular bird registration period.
```

posted by @CIKM2011 on 2011/09/15

[7] ACM Conference on Information and Knowledge Management: `http://www.cikm2011.org/`

- Hashtags can also be added loosely at the end of the tweet where the hashtag is not directly related to the sentences featured in the tweet but rather added as an additional keyword. In the example tweet below, the keywords “roma”, “nosql” and “ruby” are specified for the tweet.

```
ROMA User-Customizable NoSQL Database in Ruby
http://t.co/syJfNAHX #roma #nosql #ruby
```

posted by @nosqlweekly on 2012/02/15

Further details about hashtag characteristics and their usage can be found in Section 3.3, especially in regards to the heterogeneity of the hashtagging vocabulary and the subsequent need for a homogenization the set of hashtags used on Twitter.

Lists

Lists are an additional way of organizing message streams of other users which was introduced in 2009. A user can organize other users in lists which basically are aggregations of users who have something in common, i.e. all users of a certain list tweet about sports. The difference between simply following certain users and the addition of users to a list is that the messages of users contained in a list are not shown within the timeline. This way, the messages broadcasted by users within a list are only shown on demand, as soon as the list is clicked on. Furthermore, it is not necessary to follow users within the lists, the addition of users to a list is independent of the follower-followee connection. Also, lists may serve as a starting point for the classification of users as shown in [54] where the authors propose to characterize users based on the annotations used for the lists the users are contained in.

3.2. Twitter Data Set

This section describes the creation method and the most important features of the data set which serves as the basis for several different tasks in the course of our research. The data set is first used to analyse the hashtagging behaviour of users. More importantly, the data set serves as the basis for the recommendation of hashtags as it is exploited for the computation of hashtag recommendations. Furthermore, the data set also provides the corpus of tweets which was used for the evaluation of the proposed approach.

3.2.1. Data Set Creation

For the creation of the underlying data set, we crawled Twitter over eleven months in order to be able to create a sufficiently large data set. Generally, all tweets within the data set were obtained via the Twitter API[8] which is publicly available and provides methods enabling users to access Twitter data. In particular, the Twitter Streaming API[9] is used to gather a fraction of the public user stream of tweets. This gathered data is nearly real-time data. Generally, the Streaming API provides three different levels of access to tweets. These are named "Spritzer", "Gardenhose" and "Firehose". These three hoses differ in the volume of data which can be transferred. The entry level, Spritzer, allows for crawling roughly 1% of all public messages of users. When using Gardenhose, 10% of the public messages can be obtained and Firehose provides access to all public tweets. However, only the Spritzer is freely available to the public[10] and hence, it is the access level we were forced to use for the crawling of our data set.

For the creation of the Twitter data set, we used the `statuses/sample` API method which returns a stream of random sample of all public tweets as JSON objects. This API method allows for gathering a diverse, representative and adequate sample of tweets on Twitter. This method returns tweets as JSON objects which also contain metadata concerning the tweet, like information about the user, her profile and the content of the tweet itself. I.e. the JSON object includes information about the hashtags and the position of the hashtag within the tweet. An example JSON object of a tweet retrieved via this API can be seen in Listing A.1 in Appendix A.

Being forced to rely on the Spritzer Access level implies that the crawling of tweets is subject to restrictions regarding the volume of data retrievable via the API. The fact that on average only 12.84% of all tweets within the data sets contain hashtags at all additionally reduces the amount of crawlable data considering that only tweets which contain at least one hashtag are valuable for this data set. This is a severe constraint in regards of the crawling capabilities. In comparison to the total amount of tweets posted per day, this is a relatively small sample. However, due to the fact that the tweets returned by the API are randomly chosen, we rely on this sample to be representative and diverse.

As for the implementation of the crawling mechanisms, the crawler itself was implemented in PHP and all gathered tweets are instantly stored to the file

[8]`https://dev.twitter.com/`

[9]`https://dev.twitter.com/docs/streaming-api/methods`

[10]As of March 2012, the social media provider GNIP (http://gnip.com/) offered a stream of 10% of all tweets for $5,000 per month.

system. After having cleaned the data (e.g., from already deleted tweets which are still delivered via the API but do not feature any valuable information any more), the tweets are persistently stored to a database periodically.

3.2.2. General Aspects

For the presented data set, tweets were crawled between June 7th 2011 and May 28th 2012. In total, we were able to retrieve 386,917,626 tweets (871 GB of data) via the Twitter API. The most important facts of the crawled data set are listed in Table 3.1.

Characteristic	**Value**	**Percentage**
Crawled messages total	386,917,626	100%
Messages containing one or more hashtags	49,696,615	12.84%
Messages containing no hashtags	337,221,011	87.16%
Retweets	67,995,905	17.57%
Retweets containing one or more hashtags	14,395,494	3.72%
Direct messages, mentions	212,651,505	54.96%

Table 3.1.: Overview Tweets in Data Set

As can be seen in Table 3.1, 49,696,615 messages out of a total of 386,917,626 crawled tweets contain hashtags at all which are approximately 12.84% of all tweets. This relatively low percentage already signifies that hashtags are not well-adapted by Twitter users. 87.16% of all tweets do not feature any information about the category or content in regards to hashtags. Our analysis also showed that 17.57% of all messages are retweets (retweeted messages are later on eliminated from the data set as these would bias the evaluation results, cf. Section 3.5.2). Furthermore, 54.96% of the tweets contain one or more mentions of specific users. In this context it is important to note that these characteristics are not necessarily mutually exclusive as retweeted messages may also contain hashtags and user mentions (and vice versa). In total 3.72% of all messages are retweets which also contain at least one hashtag.

Table 3.2 summarizes the main characteristics of hashtags within the data set. In total, the data set features 7,777,194 distinct hashtags, the total number of hashtag occurrences within our data set amounts to 65,612,803. This marks an average number of 1.32 hashtags per tweet within the set of tweets containing at least one hashtag. As for the whole 386 million data set, the average number of hashtags per message is 0.16. Moreover, the hashtags featured in the data set are used 10.34 times on average whereas the median of the occurrences of

Characteristic	**Value**
Messages containing one or more hashtags	49,696,615
Hashtags usages total	65,612,803
Average number of hashtags per message	0.16
Average number of hashtags per message (within set of tweets containing at least one hashtag)	1.32
Maximum number of hashtags per message	47
Median of hashtags per message	1
Hashtags distinct	7,777,194
Hashtags occurring $\geq$ 5 times in total	757,832
Hashtags occurring $<$ 5 times in total	7,135,627
Hashtags occurring $<$ 3 times in total	6,841,523
Hashtags occurring once	5,765,835
Average number of usages per hashtag	8.43
Median number of usages per hashtag	1

Table 3.2.: Overview Hashtags in Data Set

hashtags within the data set is 1. This low median can be explained by the high number of hashtags which are only used once. This fact suggests that the distribution of hashtag popularity is a longtail distribution which is also exhibited and elaborated on in Section 3.2.3. A more detailed analysis of the hashtags and their usage within the data set is featured in the remainder of this section.

3.2.3. Hashtag Popularity Distribution

In a first step, we analysed the popularity of the different hashtags within the data set. The evaluation of the distribution of hashtags showed that this distribution follows a longtail distribution which means that only a small fraction of all hashtags used on Twitter is used at a high frequency whereas a large fraction of hashtags is used at most three times. This distribution is shown in Figure 3.2. 44 hashtags are used more than 50,000 times and hence are very popular in the Twittersphere. These 44 hashtags account to 10.69% of all hashtag usages. On average, each hashtag is used 10.34 times. Of the total 6,474,634 hashtag usages within the data set, 80% of these usages can be

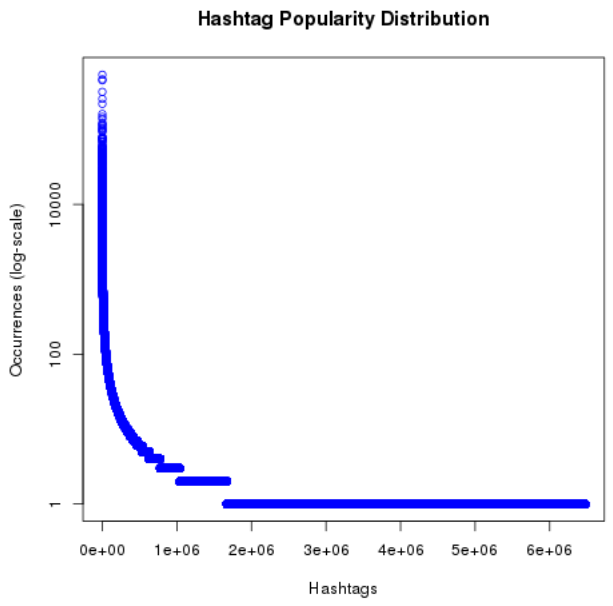


Figure 3.2.: Hashtag Popularity Distribution

accounted to hashtags which have only been used once. This is also underlined by the fact that the median usage of a single hashtag is 1. Such hashtags have been used by solely one user and often are very custom or obscure, like `#0jikai611`, `#10000ADay`, `#1000daysof2pm` or `#1509v1900`. Hence, the meaning of these hashtags can hardly be grasped by other users and therefore, such hashtags are not adapted by other users and remain used only once. Cunha *et al.* [22] detected a percentage of 60% of hashtags which are only used once within their data set. However, this set of 1.7 billion tweets was crawled within 2006 and 2009. This difference can be explained by the fact that the concept of hashtags has been introduced in 2007 and evolved within the last years. The Cunha data set contains tweets crawled already in 2006 when hashtags were not used at all and in the early stages of hashtag usage.

Such a distribution of hashtags and hence the tagging behaviour of users can be lead back to the "rich get richer" phenomenon, as already described by David Easley and Jon Kleinberg in [28]. This phenomenon basically describes that the popularity of already popular items grows stronger than the popularity of less popular items. This is also the case for hashtags where only a small fraction of hashtags are used with a high frequency. This can be lead back to the findings of Romero *et al.* [91] who showed that the process of adopting hashtags is directly related to the number of exposures to a certain hashtag (the average number of exposures until the adoption of a hashtag however varies between different topics of tweets). As Twitter users are naturally more often exposed to the most popular hashtags, they tend to adopt these hashtags and hence, these hashtags become even more popular.

Table 3.3 lists all hashtags occurring more than 100,000 times within the data set and the number of occurrences of the corresponding hashtag. From this list, we can observe that many of these hashtags are very general hashtags not necessarily describing a specific category. This generality is one of the reasons that these hashtags are very popular as they are applicable to a wide range of tweets. Many of the most popular hashtags are also very popular because of habits the Twitter community developed over the years, like the hashtags `#ff` or `#teamfollowback`.

Rank	**Hashtag**	**Occurrences**
1	`#ff`	538,795
2	`#teamfollowback`	466,186
3	`#np`	452,921
4	`#oomf`	318,190
5	`#nowplaying`	256,276
6	`#rt`	219,911
7	`#fb`	161,120
8	`#nf`	147,139
9	`#bahrain`	138,082
10	`#500aday`	121,114
11	`#followback`	119,020
12	`#bakugeki`	114,280
13	`#followme`	109,913
14	`#jobs`	104,515
15	`#tfb`	101,881

Table 3.3.: Most Popular Hashtags

In the following, the semantics of the hashtags listed in Table 3.3 are explained.

- The hashtag `#ff` stands for followfriday. This hashtag is used if a certain user wants to recommend some users she follows to her followers as she considers these recommended users to be worth following. Over time, people started to make such recommendations mostly on fridays which is the cause for the followfriday hashtag.

- The hashtags `#teamfollowback`, `#followback`, `#followme`, `#500aday` and `#tfb` are facilitated by users who want to gather more followers. Such users start to follow a large amount of users and try to motivate users to follow them back aiming at gaining more followers.
- The hashtags `#np` and `#nowplaying` are usually part of tweets which are automatically created by audio player applications. These tweets contain the title and artist of the music track the user currently is listening to.
- `#rt` is used to request the reader of the according tweet to retweet it.
- `#fb` is used for tweets which the user wants to add to her Facebook timeline automatically. All tweets containing this hashtag are automatically added to the Facebook page of the user by a specific Facebook app.
- `#oomf` stands for "one of my followers" and is typically used for tweets which are concerned with one or more followers of the tweeting user.
- `#bakugeki` is a Japanese hashtag describes a certain kind of weapon used by the characters appearing in Japanese mangas, animes and computer games.
- `#bahrain` was used during the uprise in Bahrain protesting for freedom and human rights.
- The hashtag `#jobs` is used for tweets which either contain information about job openings or people searching for a job.

Also worth mentioning is that within the top-50 hashtags in the data set, also the hashtags `#syria` and `#egypt` are featured. These two hashtags are used for tweets concerning the protests and uprisings in Syria and Egypt.

3.2.4. Hashtag Distribution per Tweet

An important fact about the hashtag usage is the number of hashtags used per tweet. As tweets are restricted to a length of 140 characters, also the amount of characters available for hashtags is limited. As for our data set, more than 80% of all tweets contained only one hashtag which can partly be explained by the limited size of tweets.

In Figure 3.3, the distribution of the number of hashtags per message is shown. From this distribution it becomes clear that most users only incorporate very few hashtags. As can be seen, about 81.87% of all tweets feature only one

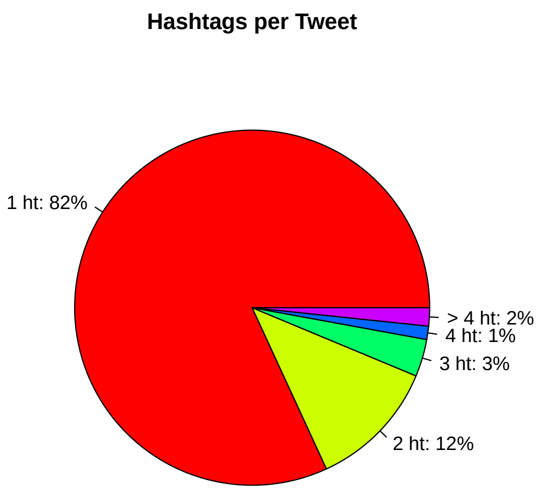


Figure 3.3.: Distribution of Hashtags per Tweet

hashtag. The share of tweets which feature two hashtags is 11.85%. The remaining 6.28% of tweets feature three or more hashtags.

Figure 3.4 once more depicts the longtail distribution of the number of hashtags per tweet. What is remarkable about this distribution is that there exist tweets containing more than 20 hashtags. This behaviour can be explained by the fact that also Twitter is subject to spam [107]. Although Twitter itself aims at recognizing spam messages and the according accounts and successfully deletes 77% of these accounts, the authors state that 17% of all spam accounts include trending topics and hashtags in their tweet in order to reach a broad audience. Hence, the more such trending topics and hashtags are used, the broader the audience. Such an approach is also known as "trend stuffing". A close examination of the relevant tweets revolves that this is exactly the case. The following tweet exemplifies such a spam tweet where unrelated but popular hashtags are used in order to propagate the tweet.

```
ugg boot shop http://uggbootshop.buyshoesale.com/
#buy #sale #price #ipad2 #iphone5 #hot #new #usa
#best #a
```

posted by @newfashioncheap on 2011-06-20

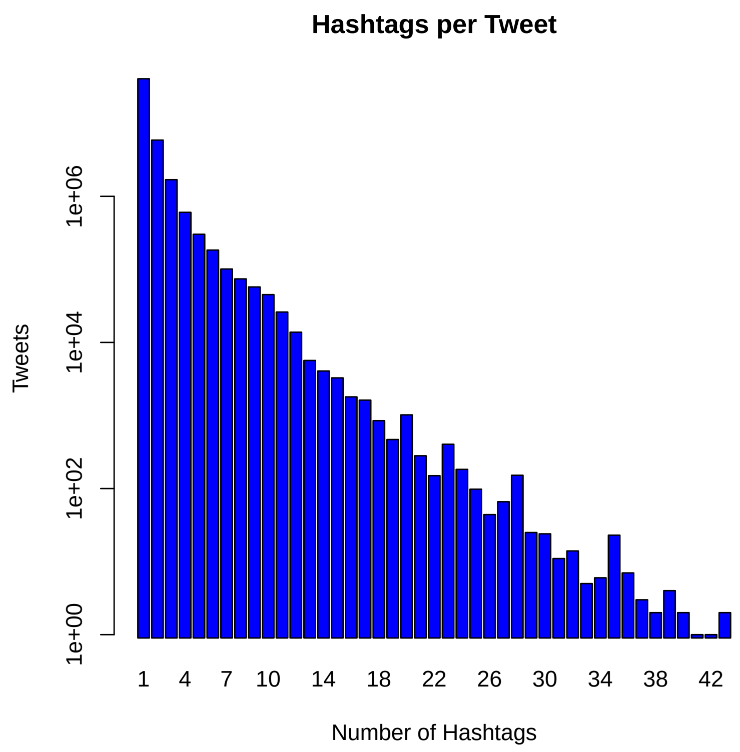


Figure 3.4.: Distribution of Hashtags per Tweet

However, also tweets seemingly not making sense contribute to such a distribution, as the following tweets exemplify:

```
#U #U #U #U #U #U #U #U #U #U #U #U #U #U #U #U #U #U #U
#U #U #U #U #U #U #U #U #U #U #U #U #U #U #U #U #U #U #U
#U #U #U #U #U #U #U x6
```

posted by @MzMiamiHeatTH on 2011-05-16

```
#Are #we #still #making #fun #of #people #who #use
#a ## #for#everything? #Or #are #they #cool #now?
#If #so, #I #love #hashtags!
```

posted by @yearofRat on 2009-07-04

3.2.5. Position of Hashtags within Tweet

Another important analysis in regards to our approach is the exact position of hashtags within the tweet. Figure 3.5 shows the distribution of the relative position of hashtag occurrences within tweets. The figure clearly shows that 12.18% of all hashtag usages occur at the very beginning of a tweet. However, 68.88% of all hashtag usages occur after 22 characters which contributes to our approach as it relies on the fact that the already entered parts of the tweet are analysed and subsequently, suitable hashtags are recommended.

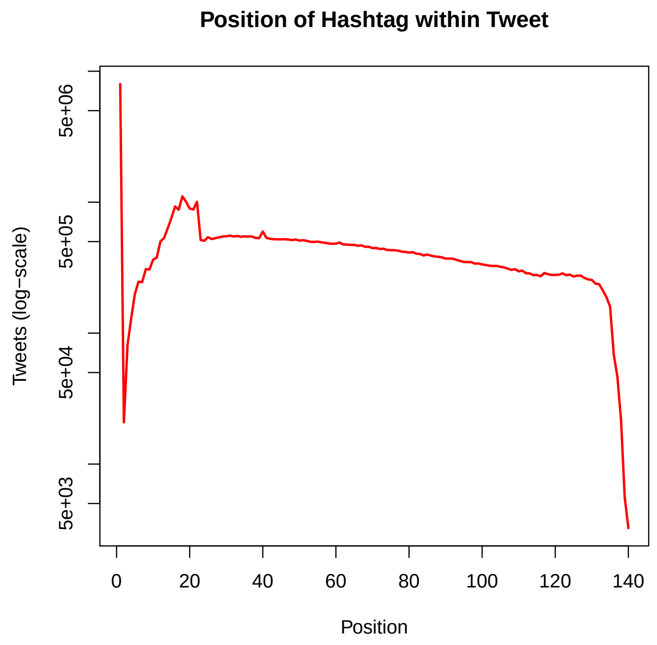


Figure 3.5.: Distribution of Hashtag Position within Tweet

3.2.6. Languages

In total, the presented data set contained tweets written in 29 different languages. Figure 3.6 show the distribution of the 12 most popular languages featured in the data set. English is the most dominant language as 64.46% of all tweets are written in English followed by 13.13 % Japanese tweets and 9.99% Spanish tweets. Portuguese, Korean, French, Russian, Indonesian, Dutch, Turkish, Italian and German are also featured within the top-12 languages used on Twitter.

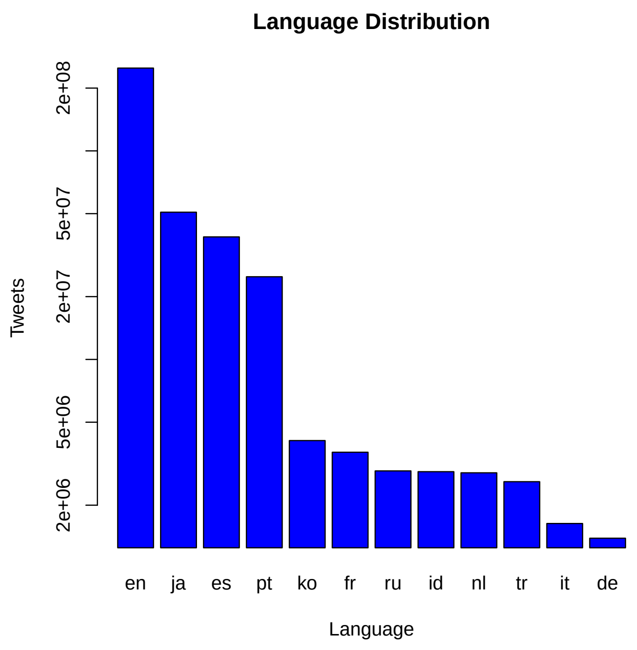


Figure 3.6.: Distribution of Top-12 Languages

3.3. In-Depth: Hashtags and Heterogeneity

Due to the immense amount of information available on Twitter and the lack of means for a manual or automatic categorization or organization of this knowledge, users of Twitter themselves introduced so-called *hashtags*. Hashtags were originally introduced by Chris Messina, a Twitter user (`@factoryjoe`). He proposed to use the hash sign as a prefix for tags for a formation of groups concerned with a certain topic or some common interest[11], as e.g. the hashtag `#barcamp` in his original proposal:

```
how do you feel about using # (pound) for groups. As
in #barcamp[msg]?
```

posted by @factoryjoe on 2007/08/23

This tweet was the starting point for a manual categorization of tweets as hashtags was taken up by the community and later also by Twitter.

[11]`http://www.nytimes.com/2011/06/12/fashion/hashtags-a-new-way-for\-tweets-cultural-studies.html`

The success of such a simple concept such as hashtags can be lead back to multiple factors:

- Hashtags are a means for **easy-to-use annotations**. The user simply has to add a hash sign to any keyword she associates with the content of the current tweet within the text of the current tweet and hence, adds information about the topic resp. the category of the tweet. The set of all hashtags of all users forms the Twitter folksonomy.
- Hashtags enable **ad-hoc group formation** as a group on Twitter can be regarded as a set of users following conversations about the same topic, like on conferences where the attendees use a hashtag related to the conference (like `#CIKM11`). This enables all attendees to **follow conversations** about the conference by simply searching for the related hashtag without having to explicitly join a certain group, like on other Social Media platforms as e.g. Facebook. Bruns *et al.* [15] describe hashtags as "a means of coordinating a distributed discussion between more or less large groups of users, who do not need to be connected through existing follower-networks". Thus, hashtags provide a means for people to follow streams of thematically connected tweets without having to follow a huge number of users in order to be able to receive all messages about a certain topic. Furthermore, such a conversation about a certain topic forms a virtual group only during the time of actual discussions as the hashtag is probably not used anymore months after a certain event has passed. Hence, due to the implicit topically focused group formation, such groups simply resolve if there are not more messages on this certain topic.

The dual role of hashtags—both the categorization of content and the creation of a virtual community—have been studied by Yang *et al.* [115]. The authors show that both roles of a hashtag are relevant for the adoption of hashtags. According to Cunha *et al.* [22], hashtags are based on the need of users to name an object or action. However, if a certain hashtag has not been used before, a new hashtag is created and hence, the heterogeneity within the hashtag vocabulary increases even if there already would have been a semantically equivalent and suitable hashtag. In [85] the authors observed the proliferation of hashtags in the course of information exchange of natural disasters (earthquakes in New Zealand and Japan) and called for means to solve this problem of heterogeneity.

Making use of hashtags aiming at a better classification of tweets is closely related to the field of automatically classifying tweets. The classification of tweets has already been addressed in research, as in [102]. This approach classifies tweets into the categories news, events, opinions, deals and private messages solely based on features which are extracted from the tweet itself, like the user, time or event information and opinion (based on a predefined list of words). However, such a classification into predefined categories cannot replace hashtags in tweets as it can only provide a coarse-grained, automatic classification of tweets whereas hashtags provide means for a very fine-grained classification of topics and thematically-focused categories. However, classification approaches may also be used to identify spam or trend-stuffing, which basically means that spammers misuse trending hashtags and terms[12] to broadcast spam messages to a big audience. The automatic classification of tweets into trend-stuffing and posts being directly related to the according trend is addressed in [50].

In a sense, hashtagging is very similar to other tagging applications, like in the areas of social bookmarking of bibliographic [69], movie databases [97] or photo categorization [99]. However, the major difference between traditional tagging systems and the hashtagging concept of Twitter is that hashtags are not only metadata, but also part of the message itself. Furthermore, traditional tags are added after the according item was uploaded or created whereas hashtags are already added to the tweet at the time of creation. Huang [48] investigated how hashtags differ from traditional tags and state that tagging on Twitter is rather used for filtering and promoting certain topics and contents in contrast to the purpose of traditional tagging, namely increasing recall of search queries. The authors consider hashtags as a way of "conversational tagging" as one tag may initiate a conversation of multiple users and also because the tag is a part of the message. The authors compared the use of tags between Twitter and Delicious (a popular social bookmarking platform[13]) and found that tags within Twitter are actively used for a shorter period of time and hence, are subject to trends.

In regards to hashtags as a means for improving search precision, Teevan *et al.* [106] investigated the difference between behavioural patterns of traditional web search and search on Twitter. They found that users make use of Twitter search as a source of information for finding timely information about news, currently trending topics and also about ongoing events. Furthermore, users facilitate Twitter search mechanisms for social information, e.g., opinions of certain users about a certain topic etc. An analysis of query logs showed that 3.25% of all Twitter queries contain user names and 21.28% of all search queries contain one or more hashtags. Furthermore, out of the 50 most popular Twitter queries (which amount to 21.19% of all queries), 50.73% of the queries

[12]The top-10 terms and hashtags mentioned within all current tweets are posted on the Twitter website and are called *trending topics*. Patterns of trends and their characteristics have been studied in [7].

[13]`http://www.delicious.com/`

contain one or more hashtags. Hence, hashtags are a frequently used and valuable means for searching and navigating through the Twittersphere and a homogeneous set of hashtags contributes to more efficient and precise search. The authors also found that Twitter queries are repeated more often and are shorter. Miles Efron came to a very similar result in his study in [30] as he discriminates two different ways of searching data on Twitter: (i) by posting questions aiming at getting other Twitter users to answer these questions and (ii) searching the already existing data on Twitter for a possible answer to the search task. For both of these tasks, the usage of a correct and popular hashtag is crucial as in the case of (i), other users (except for the followers of that certain user who automatically receive this question tweet) may only find the question if they keep track of the hashtag used. E.g., a question about a problem regarding Wordpress[14], a free blogging platform, may only be answered if it is directed to the right audience, i.e. Wordpress users and professionals. This is accomplished by using the `#wordpress` hashtag, like in the following tweet:

```
Which is the Best WordPress Translation Plugin?
http://bit.ly/LB0aHs #wordpress
```

posted by @WPMayor on 29/05/2012

As for the second way of answering information needs (that is, searching Twitter), also the usage of suitable hashtags is crucial as tweets related to a stream of conversation about a certain problem are most likely to be found by searching for a specific hashtag.

The propagation and spread of hashtags within the Twittersphere also has already been analysed. Romero, Meeder and Kleinberg analysed how users adapt hashtags they are exposed to [91]. Being exposed to a certain hashtag means that the one of the user's followees has used the according hashtag. The authors found that the speed of hashtag adoption is related to the category the original tweet stems from, i.e. politics, celebrity, technology or sports. They call this probability of adoption "stickiness". Users that are exposed to a certain hashtag multiple times are more likely to adopt a hashtag in the field of politics or sports whereas hashtags stemming from tweets about music or mentioning idioms are less likely to be adopted at the same speed. However, the frequent adoption of hashtags also contributes to a heterogeneous set of hashtags as even the most popular hashtags (i.e. those hashtags that users are most likely to be exposed to most frequently) are very heterogeneous. Consider the most popular hashtags within the data set presented in this

[14]`http://www.wordpress.com`

thesis shown in Table 3.3. Already within these very popular hashtags, many synonymous hashtags are featured. E.g., `#teamfollowback`, `#followback`, `#followme`, `#tfb` and `#500aday` are semantically equivalent. Also `#np` and `#nowplaying` share the same meaning. Thus, the set of the most popular hashtags already features semantically equivalent hashtags and the hashtag popularity distribution in Figure 3.2 shows that a large portion of hashtags are used less than three times. Bruns and Burgess showed in [15] that hashtags are also homogenized manually by the community. This is done by replying to users who make use of hashtags which are synonymous to the widely used hashtag and pointing users to the more popular hashtags. Such an approach is a very tedious task and requires a considerable amount of human effort. A tweet exemplary for such user behaviour can be seen in the following:

```
#earthquake Apparently the official hashtag is
#eqnz; #doingitwrong. Glad someone's got priorities
sorted.
```

posted by @adzebill on 2010-09-04

In this thesis, we propose recommender system for hashtags which provides Twitter users with hashtags highly suitable for the tweet the user is currently entering.

3.4. Hashtag Recommendation Concept

We propose a hashtag recommender system which aims at providing microblog users with highly suitable recommendations of hashtags for the tweet the user currently enters. By facilitating such an approach, the problem of synonymous hashtags can be dealt with already at the time of the creation of a tweet as users are provided with recommendations of hashtags which have already been used by other users entering a semantically similar tweet. Hence, synonymous hashtags are not even created. Furthermore, by providing suitable hashtags, users are encouraged to enter hashtags at all aiming at increasing the number of tweets containing hashtags. In this section, we present a concept for hashtag recommendations based on a crawled data set.

During the input of a certain tweet, recommendations have to be computed on the fly (with every keystroke) in order to ensure that the user is provided with the most suitable hashtags for his tweet. Another reason for the computation of hashtags already during the insertion of a microblog entry is that hashtags may appear at any position within a microblog entry, e.g. also embedded in the actual text, like the hashtag `#crowdsourcing` in the following tweet:

``Better shred than read: DARPA uses competitive #crowdsourcing to revive destroyed documents http://t.co/A7zOWass''

Posted by @abellogin on 2012/01/12.

Generally, we split the computation process into the following main steps:

1. For the given tweet (or parts of it), retrieve the most similar tweets within the tweet data set.

2. Extract the hashtags contained in the set of the most similar tweets.

3. Rank the extracted hashtags according to some ranking algorithm.

4. Recommend the top-k hashtags to the user.

These steps are described in detail within the next sections.

3.4.1. Basic Algorithm

In the following, the recommendation algorithm for the computation of hashtag recommendations for microblog platforms is explained. Figure 3.7 also depicts the workflow of the recommendation process. For all of the following steps, the data set used is the data set presented in Section 3.2, which is stripped from all tweets not containing any hashtags as these entries do not contribute to the computation of hashtag recommendations.

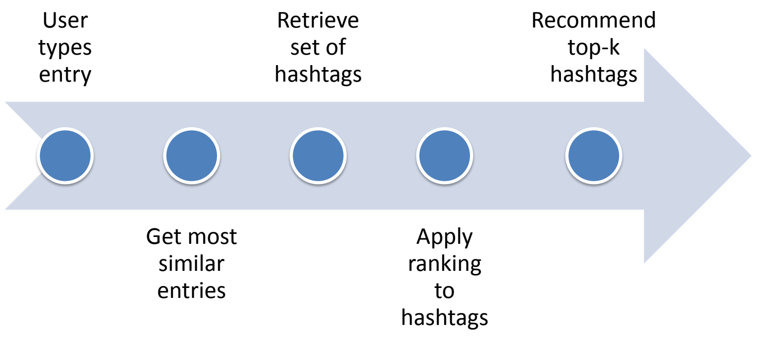


Figure 3.7.: Basic Computation of Recommendations—Workflow

Algorithm 1 depicts the detailed algorithm underlying the computation of hashtag recommendations. After the given initialisation steps, the most similar entries are searched for within the tweet data set. This is realized by making use of a fulltext index which is responsible for retrieving the set of top-n most similar entries for a given input entry τ_i and a given similarity measure $similarityMeasure$. For accomplishing this task, each entry $\tau_j \in C$ within the data set is compared to the input entry τ_i and the similarity measure determines the similarity score for all such pairs (τ_i, τ_j) of entries. Further information about the different similarity metrics used for our approach can be found in Section 3.4.2. The entries within the set of the most similar entries are sorted and the top-n entries ($entryCand$) within this sorted list are used as entry candidates for the next step. The so-called hashtag recommendation candidates are extracted such that all hashtags occurring in each entry $\tau \in entryCand$ are extracted and added to the set $hashtagCand$. This unordered set is subsequently ranked based on a given ranking method. Our proposed ranking mechanisms are described in Section 3.4.3. Again, this list of hashtag recommendation candidates is cut off in order to retrieve the top-k most suitable hashtag recommendations. The final list of ranked hashtag recommendation candidates is then the result of the recommendation computation process and presented to the user.

3.4.2. Similarity of Microblog Entries

The hashtag recommendation candidates for a certain input entry τ_i are extracted from the entries most similar to τ_i. This is due to the fact that we assume that entries containing similar contents also contain similar hashtags. Hence, a function describing the similarity (or distance) of entries is required. Therefore, we chose to evaluate different traditional functions aiming at finding the most suitable similarity measure in regards to the quality of the subsequently computed hashtag recommendations.

Generally, string similarity measures can be categorized into set (or bag)-based and vector-space models [72]. Set-based models rely on a bag-of-words approach where all terms of a given text form a bag of words, i.e. the order in which the terms originally appeared in is ignored. The computation of the similarity of two given strings is computed based on the bags of terms stemming from the two input strings and is mostly related to the intersection of these two bags. E.g., the two strings “Federer wins over Nadal at Wimbledon” is equal to “Nadal wins over Federer at Wimbledon” as the two bags for the two texts are equal as the intersection of these two strings is equivalent to both of the entries. In contrast, vector-space models rely on the representation of texts as (mostly weighted) vectors where each dimension of the vector corresponds to one term within the corresponding document. A weighting of the dimensions is

```
Data: $C$, collection of all microblog entries within the reference tweet
      data set
Data: $\tau_i$, input microblog entry
Data: $similarityMeasure$ used for candidate entries
Data: $rankingMethod$ used for hashtag recommendations
Data: $n$, number of similar hashtags used for recommendation
Data: $k$, number of hashtags recommended
Data: $\delta$, threshold minimum similarity score
Result: Ranked list of top-$k$ hashtag recommendations
begin
    // Initialisation
    similarEntries, hashtagCand := {}
    sortedEntryList, entryCand, rankedCandList, resultList := [ ]
    simScore := 0.0
    // (1) Find most similar microblog entries
    foreach Entry $\tau_j$ within $C$ do
        simScore = similarity($\tau_i$, $\tau_j$, similarityMeasure)
        if $simScore > \delta$ then
            similarEntries = similarEntries $\cup$ {($\tau_j$, simScore)}
        end
    end
    // Sort entries w.r.t. similarity score
    sortedEntryList = sort(similarEntries)
    // Use top-$n$ most similar entries
    entryCand = cutOffList(sortedEntryList, n)
    // (2) Extract hashtags
    foreach Entry $\tau_j$ within entryCand do
        hashtagCand = hashtagCand $\cup$ extractHashtags($\tau_j$)
    end
    // (3) Perform ranking of hashtags
    rankedCandList = rank(hashtagCand, rankingMethod)
    // (4) Compute top-k recommendations
    resultList = cutOffList(rankedCandList, k)

    // Return top-k recommendations
    return resultList
end
```

Algorithm 1: Basic Recommendation Algorithm

performed in order to weigh down terms which e.g. occur at a high frequency in many documents of the corpus and may not be significant in regards to describing the characteristics of a certain document. Besides these two types of exact similarity measures, we also evaluated the suitability of approximate string matching techniques (also known as fuzzy string matching), in particular we evaluated the Levenshtein distance.

In the following, we present both distance and similarity functions for two entries τ_i and τ_j. Hence, these two directly opposed functions have to be transformed into one uniform notation describing how similar (or different) two strings are. Two strings are equal if either the similarity function computes to 1 or the distance functions computes to 0. However, it is easily observed that a distance function can be turned into a similarity function (see Equation 3.1) and vice-versa, a similarity function can be turned into a distance function (see Equation 3.2).

$$sim(\tau_i, \tau_j) = 1 - dist(\tau_i, \tau_j) \tag{3.1}$$

$$dist(\tau_i, \tau_j) = 1 - sim(\tau_i, \tau_j) \tag{3.2}$$

All of the presented measures are normalized to the range $[0, 1]$ in order to make the similarity computation results comparable.

In the following, we present the different similarity (resp. distance) functions we chose to evaluate for the hashtag recommendation task. We first present two set-based similarity functions (Jaccard and Dice similarity coefficients), then the cosine similarity measure based on the vector space model and according weighting functions and finally the Levenshtein distance as a representative for approximate string matching.

Jaccard Similarity Coefficient

The Jaccard similarity coefficient is a set-based similarity measure. This similarity coefficient defined as the fraction between the overlap of terms and size of the union of the two entries τ_1 (query) and τ_2 (document) which are to be compared. Equation 3.3 shows the computation of the coefficient, where T_i and T_j are the respective sets of words contained in the corresponding entries τ_i and τ_j.

$$sim(\tau_i, \tau_j) = \frac{|T_i \cap T_j|}{|T_i \cup T_j|} \tag{3.3}$$

Consider the following example entries:
τ_i = "Roger Federer is about to win against Rafael Nadal in Wimbledon",
τ_j = "I wish Rafael Nadal would do better against Roger Federer in Wimbledon".

The intersection of the two sets of tokens based on the texts is 7 ($\tau_i \cap \tau_j =$ {Roger, Federer, Rafael, Nadal, Wimbledon, against, in}). The size of the union of the tokens of the two texts ($|\tau_i \cup \tau_j|$) is 16 and hence, the Jaccard similarity coefficient accounts to 0.44.

The Jaccard coefficient may also be modelled in the (boolean) vector space (as in [120, 96]). A boolean vector D representing a document d (in particular, an entry τ containing terms t_i) is defined as

$$D = (D_i)_{1<i<|\mathcal{T}|} = \begin{cases} 1 & if\ term\ t_i \in \mathcal{T} \\ 0 & else \end{cases} \tag{3.4}$$

where $\mathcal{T}$ is the set of all distinct terms within the corpus and t_i are the terms occurring in this set (document). Based on two such vectors, one representing the query and another vector representing the document which is to be compared, Jaccard similarity can be defined as

$$sim(\tau_i, \tau_j) = \frac{\sum_{t \in T_{\tau_i,\tau_j}} (D_{\tau_i} \cdot D_{\tau_j})}{|\tau_i| + |\tau_j| - \sum_{t \in T_{\tau_i,\tau_j}} (D_{\tau_i} \cdot D_{\tau_j})} \tag{3.5}$$

where D_{τ_i} and D_{τ_j} are the boolean vectors representing two entry documents τ_i and τ_j, T_{τ_i,τ_j} is the set of all distinct terms occurring in both τ_i and τ_j and $|\tau_i|$ resp. $|\tau_j|$ is the document length of the according document.

Dice Similarity Coefficient

The Dice coefficient also is a set-based similarity function and is very similar to the Jaccard similarity coefficient. Dice also defines the similarity in regards to the similarity of the set-interpretation of both the query string τ_1 and the actual document τ_2. The function is defined as the fraction between the overlap of terms contained in both the query entries τ_i and an entry τ_j and the sum of the lengths of the two documents (microblog entries). The coefficient can be computed as in Equation 3.6, where T_i and T_j are the respective sets of words contained in the corresponding entry τ_i and τ_j.

$$sim(\tau_i, \tau_j) = \frac{2 \cdot |T_i \cap T_j|}{|T_i| + |T_j|} \tag{3.6}$$

Consider the example texts already mentioned the description of the Jaccard similarity coefficient (cf. Section 3.4.2). Again, the intersection of the two sets is 7. However, in the case of Dice similarity coefficient, the denominator of the fraction is defined as the sum of the cardinalities of the two sets. In the case of this example, $|\tau_i| = 11$ and $|\tau_j| = 12$. Hence, the denominator amounts to 23 and the resulting Dice similarity coefficient is 0.30.

Similarly to the vector space implementation of the Jaccard similarity coefficient, the Dice similarity coefficient can also be modelled in the vector space:

$$sim(\tau_i, \tau_j) = \frac{2 \cdot \sum_{t \in T_{\tau_i,\tau_j}} (D_{\tau_i} \cdot D_{\tau_j})}{|\tau_i| + |\tau_j|} \tag{3.7}$$

where D_{τ_i} and D_{τ_j} are the boolean vectors representing two microblog documents τ_i and τ_j, T_{τ_i,τ_j} is the set of all distinct terms occurring in τ_i and τ_j and $|\tau_i|$ resp. $|\tau_j|$ is the document length of the according document.

For equal input texts, the Dice similarity coefficient is smaller or equal to the Jaccard similarity coefficient. This is due to the fact that the denominator for Dice is greater than the denominator for Jaccard. Dice and Jaccard are only equal in the case of an empty intersection of the set of the tokens of the two input texts. In any other case, the Dice coefficient is smaller than the Jaccard coefficient, per definition.

Cosine Similarity

The vector space cosine similarity function is one of the predominant measures in information retrieval and is widely used for computing query-document and document-document similarities [32]. It is based on the (weighted) term vectors of both the query and the respective document which is to be compared to the query. In the case of searching for the best matching entry for a certain input entry, the query is defined by the input entry and the cosine similarity is computed for the input entry and every single entry contained in the database. The definition of cosine similarity between vectors (i.e. the cosine of the angle between these two vectors) is defined as shown in Equation 3.8 where v_i is the vector representing the input entry (the query) and v_j is vector representing the reference entry from the reference database.

$$cos(v_i, v_j) = \frac{v_i \cdot v_j}{\|v_i\| \, \|v_j\|} = \frac{\sum_{k=1}^{N} v_{k,i} \cdot v_{k,j}}{\sum_{k=1}^{N} v_{k,i}^2 \sum_{k=1}^{N} v_{k,j}^2} \tag{3.8}$$

Closely related to the cosine similarity of vectors is the weighting of terms [93] which allows for the computation of weight factors of each single term featured in the vectors. We chose to evaluate the cosine similarity of weighted term vectors computed by two different weighting schemes: the term frequency-inverse document frequency weighting scheme and the BM25 Okapi weighting scheme, which are defined below.

The traditional *term frequency-inverse document frequency* (tf-idf) weighting scheme aims at estimating the relevance and importance of a certain term in relation to the whole document corpus (in our case the reference database).

This function is based on two components: term frequency (*tf*) and inverse document frequency (*idf*). *tf* can be defined as the number of occurrences of the given term t_i within the current document D (in particular, a microblog entry). The *idf* component basically defines how relevant a term is in relation to the whole set of documents, as can be seen in Equation 3.9 where N is the total number of documents within the reference database and $n(t_i)$ is the number of documents which contain the given term t_i.

$$idf(t_i) = log\frac{N}{n(t_i)} \tag{3.9}$$

These two components are then combined to the tf-idf weight of a given term as can be seen in Equation 3.10.

$$\textit{tf-idf}(t_i, D) = tf(t_i, D) \cdot idf(t_i) \tag{3.10}$$

The second weighting scheme we made use of is the *BM25 Okapi* weighting scheme [90] which additionally incorporates the average length of a document. Furthermore, smoothing and tuning factors are incorporated. BM25 is computed based on a similar definition of the inverse document frequency as can be seen in Equation 3.11, where $n(t_i)$ again is the number of documents which contain the given term t_i and N is the total number of documents within the reference data set.

$$idf(t_i) = log\ \frac{N - n(t_i) + 0.5}{n(t_i) + 0.5} \tag{3.11}$$

The BM25 Okapi weight of a given term in relation to the reference data set can subsequently be computed as in Equation 3.12, where D is the total number of documents in the reference data set, $f(t_i, D)$ is the number of occurrences of term t_i in the document D, $|D|$ is the length of document d and $avgLen$ is the average length of all documents within the reference data set. Furthermore, two tuning parameters are used: k_1 is set to 1.2 and b is set set 0.75 as proposed in [90].

$$bm25(t_i, D) = idf(t_i) \cdot \frac{f(t_i, D) \cdot (k_1 + 1)}{f(t_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgLen})} \tag{3.12}$$

Levenshtein Distance

The Levenshtein distance [67] (also known as edit-distance) is a traditional lexical distance measure for strings. The Levenshtein distance is particularly widely used for spelling corrections as it provides for a simple means for approximate string matching (e.g. [70]). It is defined by the minimum number of edits required for transforming string t_i into string t_j. In this context, edits may be (i) the insertion of a character, (ii) the deletion of a character or (iii) the replacement of a certain character with another character. I.e. the Levenshtein distance between the words “house” and “mouse” is 1 as only one edit (namely the replacement of the first character) is required.

In particular, we applied the Levenshtein similarity measure as shown in Equation 3.13, where the number of edits required to turn one document into the other ($lev(\tau_i, \tau_j)$) is divided by the minimum of the lengths of the two documents. This allows for a normalization of the result of the similarity function to the range of $[0, 1]$, which is highly desirable for the subsequent ranking of entries and which makes the different similarity functions comparable.

$$sim(\tau_i, \tau_j) = 1 - \frac{lev(\tau_i, \tau_j)}{min(|\tau_i|, |\tau_j|)} \tag{3.13}$$

One variant of the Levenshtein distance we also evaluated, is the Levenshtein-Damerau distance [23], which also allows for transpositions (besides insertions, deletions and replacements). E.g., the distance of the two strings “huose” and “house” is 1 as two adjacent characters have to be transposed, whereas with the traditional Levenshtein distance, the distance would have been 2.

3.4.3. Ranking Strategies

The computation of the most similar entries for the given input entry τ_i (as described in Section 3.4.2) results in a set $\mathcal{S}_{\tau_i}$ of similar entries for the given input entry. The set of hashtag recommendation candidates $\mathcal{H}_{\tau_i}$ is extracted from the entries within $\mathcal{S}_{\tau_i}$. The hashtags featured in the set $\mathcal{H}_{\tau_i}$ have to be ranked according to some relevance criteria. The ranking of hashtag recommendation candidates is a crucial step in regards to the performance of the recommender systems. The user is provided with the top-k recommendations, where k denotes the number of hashtags contained in the set of hashtag recommendation candidates. Hence, the most appropriate hashtags for the given entry have to be ranked as high as possible in order to provide the user with a satisfactory set of recommendations. Usually, a set of 5–10 recommendations is most appropriate which also corresponds to the capacity of short-term memory [73]. Also, microblog entries are a means for fast communication and hence, the list of hashtags appropriate for a given entry has to be easily and rapidly comprehensible for users who are just entering an entry which strengthens the fact that only a few recommendations are provided. Furthermore, the limited space available for displaying recommendations additionally constrains the number of hashtags to be recommended. The work by Bollen *et al.* [11] underlines this choice as the authors conducted an experiments showing that presenting users with a large number of good and valuable recommendations is counterproductive as the choice of a recommendation becomes inherently difficult for the user. The experiments showed that presenting a set of recommendations containing between 5 and 20 items limits the problem of choice overload while at the same time offering variability to the user.

In the following, we present the proposed ranking strategies which rely on the content of the entries, overall hashtag usage statistics and also timely features.

SimRank

This ranking method is based on the similarity of the input entry τ_i and the entries contained in the set of the most similar entries $\mathcal{S}_{\tau_i}$. This ranking hence resembles a recommendation of hashtags based on the entries most similar to the input entry and therefore is directly related to the message content. For the ranking, the similarity values computed by the similarity measures presented in Section 3.4.2 are directly used for the ranking of the hashtag candidates such that the higher the similarity of the input entry τ_i and the entry featuring the according hashtag, the higher the ranking of the hashtag. If a hashtag is featured in multiple entries, the highest score obtained is used for the ranking.

RecCountRank

The recommendation-count rank is based on the popularity of hashtags within the hashtag recommendation candidate set. This implies that the more similar messages contain a certain hashtag, the more suitable the hashtag might be. This ranking follows the "one person, one vote" principle introduced by Hill and Terveen [43], which was also facilitated by Chen *et al.* [19] for the ranking of URL recommendation candidates extracted from Twitter. Based on this principle, each entry within the set of most similar entries is able to vote for each of the hashtags. I.e. if a certain hashtag is contained in the entry, the hashtag is voted. These votes are counted and based on the count values, the ranking is performed such that the more votes a hashtag obtained, the higher is its ranking.

$$RecCountRank(\mathcal{S}_{\tau_i}, h) \ = | \ \{\tau_j \ | \ h \in \tau_j \wedge \tau_j \in \mathcal{S}_{\tau_i}\} \ | \tag{3.14}$$

GlobalPopularityRank

The so-called global popularity rank is based on the global popularity of hashtags within the whole underlying data set $\mathcal{S}$, i.e. the set of all hashtags. Again, a "one person, one vote" approach is used, as in RecCountRank. In contrast, the set of all entries containing hashtags is used as input for the vote. As only few hashtags are used at a high frequency, it is likely that such a popular hashtag matches the user's entry. Therefore, ranking the overall most popular hashtags from within the candidate set higher is also a suitable approach for the ranking of hashtags.

$$GlobalPopularityRank(\mathcal{S}, h) = \mid \{\tau_j \mid h \in \tau_j \wedge \tau_j \in \mathcal{S}\} \mid \tag{3.15}$$

MostRecentlyUsedRank

Besides considering either the content of the entry corresponding to a certain hashtag or the popularity of the given hashtag for the ranking of the hashtags, also timely features may play an important role for the ranking of hashtags. This is especially the case for highly dynamic microblogging platforms where trending topics evolve rather quickly and hence, the according hashtags may also gain popularity quickly. Therefore, we propose two different ranking methods based on timely features of the entries and according hashtags.

The MostRecentlyUsedRank considers the recency of the usage of the hashtag recommendation candidates. The more recent a certain hashtag has been used, the higher its ranking. This ranking enables the detection and prioritization of currently trending hashtags (most probably related to trending topics) which have been used recently. In particular, the ranking is based on the timespan between the time of the computation of recommendations and the time the hashtag was used the last time. The computation is performed on the according timestamps.

$$MostRecentlyUsedRank(\mathcal{S}_{\tau_i}, h) = min(\{now() - createdAt(\tau_j) \mid \tau_j \in \mathcal{S}_{\tau_i}\}) \tag{3.16}$$

AvgUsageAgeRank

The second time-related ranking method is AvgUsageAgeRank. This ranking method aims at identifying how “trending” a certain hashtag is, i.e. how often it has recently been used. Therefore, we propose to make use of the average timespan between the time of recommendation and the respective usages of a certain hashtag. I.e. the more often the hashtag has been used recently, the higher is its ranking as the hashtag can be regarded as recently trending.

$$AvgUsageAgeRank(\mathcal{S}_{\tau_i}, h) = avg(\{now() - createdAt(\tau_j) \mid \tau_j \in \mathcal{S}_{\tau_i}\}) \tag{3.17}$$

Performing the ranking of hashtags solely based on the most recently used hashtags or the average recency of usage is not likely to gain satisfactory results. However for hybrid ranking methods, adding recency and timely factors to the ranking may contribute to a better result as trends within the Twittersphere are also taken into account.

Hybrid Ranking

Besides the previously presented basic ranking algorithms, we propose to use hybrid ranking methods which are based on the presented basic ranking algorithms. The combination of two ranking methods is computed by the following formula:

$$hybrid(r1, r2) = \alpha \cdot r1 + (1 - \alpha) \cdot r2 \tag{3.18}$$

where α is the weight coefficient determining the weight of the respective ranking within the hybrid rank. $r1$ and $r2$ are normalized to be in the range of $[0, 1]$ and can therefore be combined to a hybrid rank.

3.4.4. Co-Occurrence of Hashtags

In order to further enhance the quality of recommendations, we propose to also make use of a co-occurrence analysis of hashtags. Consider the following microblogging message which contains four hashtags:

```
garshol.priv.no/blog/231.html RDF triple stores
- an overview by @larsga ... simply, brilliant!
#linkeddata #datastore #nosql #rdf
```

posted by @mhausenblas on 19/09/2012

By performing an analysis of co-occurrences of hashtags (that is, multiple hashtags appearing in the same microblogging messages), hashtag recommendations may be enhanced. Hashtags related to either the hashtags the user has already used in the current message or hashtags related to the recommended hashtags may be added to the set of recommendation candidates. If a user e.g. specified the hashtag `#rdf`, he may also be presented with the hashtags `#linkeddata` or `#nosql` based on an analysis of co-occurring hashtags.

Modelling Co-Occurrence

In particular, we propose to model hashtags co-occurring within a given tweet as pairs of hashtags $\{hashtag1, hashtag2\}$. In the case of three or more co-occurring hashtags, these are splitted into distinct pairs of hashtags. Such a model features multiple advantages: (i) pairs allow for a better optimization in regards to computing recommendations (ii) the evaluations of the data set showed that only 6.28% of all messages feature more than two hashtags, hence one pair is sufficient for storing co-occurring hashtags for the majority of the tweets and (iii) this approach enables a more fine-grained computation of hashtag recommendation candidates. As co-occurrence of hashtags within a single microblogging message is a symmetrical property, we chose to use unordered pairs such that we do not have to store both pairs $(hashtag1, hashtag2)$ and

($hashtag2, hashtag1$). In order to be able to assess how often two hashtags are on the same microblogging message throughout the corpus, we extend this pair of co-occurring hashtags with a count value c representing the number of co-occurrences across the underlying data set. Hence, all co-occurring hashtags are modelled as triples (x, y, c).

The actual selection of recommendation candidates is performed by selecting all triples which feature one of the input hashtags. As for the input of the co-occurrence analysis to be conducted, two different approaches can be taken into account:

- The set of hashtags already specified for the current microblogging message can be used as input for the computation of further recommendation candidates.
- The set of hashtags candidates already computed by the basic algorithm as defined in Section 3.4.1 can be used as input in order to detect further hashtag recommendation candidates.

Co-Occurrence Analysis

Based on the previously presented notation of co-occurrence of hashtags, we performed an analysis for the data set presented in Section 3.2. In the course of this evaluation, we extracted all pairs for tweets which contain more than one hashtag. By doing so, we created a set of 20,663,292 ordered pairs which we reduced to 10,331,646 unordered pairs and the according count values.

Table 3.4 contains the results of the analysis. As can be seen, 11 pairs of hashtags occur on more than 50,000 messages (in a data set of a total of 49,696,615 messages) which amounts to 0.1% of all messages (pair popularity). However, the majority of pairs occurs at a low frequency. For 97.85% of all pairs, the two hashtags of the respective pairs only co-occur within less than 10 messages. These numbers indicate a long-tail distribution of pairs. Hence, a large fraction of the computed pairs occur infrequently. Therefore, making use of such unpopular pairs for the computation (or enhancement) of hashtag recommendations may lead to a more diverse and homogeneous set of hashtags as combinations of hashtags which only very few users made use of are considered for recommendation. Therefore, we propose to only consider those hashtags stemming from co-occurrence analysis having a count-value higher than a certain threshold.

No. of Pairs	**% of Pairs**	**Pair Popularity**
11	$1.1 \cdot 10^{-6}\%$	$> 50,000$ ($1 \cdot 10^{-3}\%$ of all entries)
160	$1.5 \cdot 10^{-5}\%$	$> 10,000$ ($2 \cdot 10^{-4}\%$)
300	$2.9 \cdot 10^{-4}\%$	$> 5,000$ ($1 \cdot 10^{-4}\%$)%
1,349	$1.3 \cdot 10^{-3}\%$	$> 1,000$ ($2 \cdot 10^{-5}\%$)%
18,087	$1.7 \cdot 10^{-2}\%$	> 100 ($2 \cdot 10^{-6}\%$)%
10,313,356	99.82%	< 100 ($2 \cdot 10^{-6}\%$)
10,109,704	97.85%	< 10 ($2 \cdot 10^{-7}\%$)

Table 3.4.: Co-occurrence Analysis

Ranking

As for the ranking of hashtag recommendation candidates computed by the above presented approach of co-occurrence analysis, we propose two different methods:

- A stand-alone ranking method which can be combined into a hybrid rank (see Section 3.4.3 for a description of the hybrid ranking method). For this ranking method, we propose to make use of the count-value c stemming from the co-occurrence pair (x, y, c) where either x or y is an input hashtag, i.e. we rely on the popularity of a certain rule for the ranking of the respective recommendation candidates. This approach enables an integration of the computed recommendation candidates with other recommendation candidates stemming from the basic algorithm for hashtag recommendations.
- The analysis of co-occurrences can also be exploited to enhance a ranking of hashtags already computed by one of the ranking methods presented in Section 3.4.3, i.e. no further recommendation candidates are added. Instead, we propose to use the hashtags computed by a co-occurrence analysis to boost the ranking of those hashtags which also have been computed by the basic algorithm for hashtag recommendations. Such a boosting can be implemented by multiplying the already computed rank by a boosting factor β which resembles the ranking of the hashtag within the set of hashtags computed by a co-occurrence analysis.

3.4.5. User-Specific Recommendations

For the hashtag recommendation concept presented in this dissertation, we also propose to make use of the user's hashtagging behaviour history, i.e. to analyse the hashtags the user previously made use of in her microblogging messages. By facilitating such an approach, the hashtag recommendations can be customized to the user's previous hashtagging behaviour. This may lead to a higher acceptance rate of recommendations as users tend to re-use hashtags once they adopted these hashtags. This additional enhancement of recommendations can be implemented by introducing a boost value β representing e.g. the number of usages of the respective hashtag. This value is used to boost the rank of all hashtags within the set of recommendation candidates which the user already made use of in the past.

3.5. Evaluation

The following section describes the evaluation of the proposed algorithms.

The hashtag recommendation task fulfilled by the proposed system can be seen as a recommendation of good items according to [39]. Gunawardana *et al.* state that the recommendation of good items to users is basically concerned with providing users with recommendations for items the user is likely to accept. In the case of the proposed system, the user is presented with hashtags which are most likely to be useful to the user and can hence be identified as the recommendation of good items. More specifically, Gunawardana *et al.* identify two major subtasks: (i) the recommendation of some good items and (ii) the recommendation of all good items. Such a distinction between recommendation tasks can also be found in [42].

The recommendation of all good items aims at providing the user with all important and suitable items. Hence, such an approach results in a long list of recommendations. As for our hashtag recommendation system, such an approach is not feasible simply due to the fact that a tweet is restricted to a maximum length of 140 characters which significantly limits the number of hashtags which can be used within a tweet. A second restriction which implies that the task cannot be identified as the recommendation of all good items is that Twitter is a very fast and efficient medium. Hence, the user can only be presented with a short list of possible hashtag candidates as the goal is to be able to quickly post tweets including the selection of one or more presented and recommended hashtags. Thus, there is no point in recommending all suitable hashtags to the user. Therefore, our evaluation (and also the presented recommendation approach) aims at recommending some items (namely hashtags) to the user. Jannach *et al.* refer to this task as a classification task which aims at recommending the most relevant items to a user [51].

We chose to perform an automatic evaluation to assess the quality of the computed recommendations. The main focus of the evaluation lies on assessing the accuracy of the computed recommendations. This is due to the fact that such an automated offline evaluation is able to assess the quality of the algorithms fast and efficiently. Such offline evaluations can be implemented based on single evaluation runs which incorporate millions of tweets and the according hashtags automatically. Also the tuning of parameters within the algorithms can be realized efficiently as every change in parameters can immediately be evaluated when using offline evaluation methods. Moreover, it is hardly feasible to gather a sufficient number of test users as the test users would have to evaluate all proposed algorithms in multiple different configurations.

I.e. we based our evaluation on historic tweets contained in the crawled data set and evaluated whether our recommendation algorithms would have recommended hashtags that the users already made use of within their tweets. Hence, the evaluation assesses whether our recommender system would have reconstructed the hashtags of the original tweet. The results of such an evaluation can be seen as a baseline as the system might even have recommended better suitable hashtags which were not featured in the original tweet and hence, were evaluated to not be correct. However, assessing whether a certain hashtag would have suited the given tweet better can hardly be detected automatically as this decision depends on various influence factors like the popularity of the hashtag, the community using this hashtag, the fact that hashtags are adopted from followers and followees, the stream of conversation evolved around this hashtag, etc. Therefore, we rely on an automatic evaluation relying on huge numbers of tweets and consider the results of this evaluation as baseline results. Still, the development of a client for future user studies is part of ongoing work.

3.5.1. Metrics

Based on the definition of [39] and adapting it to the task of recommending hashtags, the possible outcomes of the recommendation of a hashtag are shown in Table 3.5. True Positives are items (hashtags) which were recommended and also actually selected[15] by the user whereas False Positives were recommended, however not picked by the user. False Negative samples are hashtags which were not recommended but would have been suitable for the user and True Negatives are items neither recommended nor picked.

The exact evaluation procedure and the corresponding algorithm can be found in Section 3.5.3. The fact that only historic information, namely the actual use of hashtags within tweets, form the basis for the evaluation, an evaluation of False Positives and True Negatives is not feasible as only accepted and used

[15]The notion of “selecting” an item refers to having used the item in the original tweet in our evaluation setup.

	recommended	not recommended
picked	True Positive	False Negative
not picked	False Positive	True Negative

Table 3.5.: Recommendation Confusion Matrix

hashtags are recorded. Therefore, only True Positive and False Positive items can be considered for the computation of evaluation metrics. Hence, the result of evaluating a single recommendation can only be either 0, if the user did not accept a recommendation or 1 if the user accepted the recommended item. Based on these definitions, the two evaluation metrics precision and recall can be computed as shown in Equations 3.19 and 3.20. These metrics originate from the area of information retrieval and are traditionally also used for the evaluation of the accuracy of recommender systems [25]. These metrics have originally been proposed by Cleverdon and Kean [20].

$$precision = \frac{|\#TruePositives|}{|\#TruePositives + \#FalsePositives|} \tag{3.19}$$

$$recall = \frac{|\#TruePositives|}{|\#TruePositives + \#FalseNegatives|} \tag{3.20}$$

Precision defines a ratio between the number of actually accepted recommendations (#TruePositives) and the total number of recommendations provided to the user. This total number of recommendations is computed by summing up the number of accepted (#TruePositives) and the number of neglected recommendations (#FalsePositives). Hence, the precision characterizes the amount of correctly computed recommendations regardless of how many more items would have been correct. In contrast, the *recall* measure defines a ratio between the accepted recommendations and the total number of correct recommendations possible (all hashtags originally used). Due to the historic evaluation, the set of applicable hashtags is defined by the hashtags the user originally used within a tweet.

It is important to note that precision and recall are inversely related. This fact that implies that if the number of recommended items is increased, the recall value also increases. In contrast the precision values deteriorates as less recommended items are correct and hence the precision of recommendations decreases.

The third measure used is the F1-score which basically combines the precision and recall values to one single measure. This is very beneficial as the accuracy and performance of a recommender system can only be described by both of these measures, they cannot be interpreted independent from each other. The according formula of the F1-score can be seen in Equation 3.21. We also make use of this measure throughout our evaluation as it can be seen as a more universally comparable measure for recommendation algorithms [51].

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3.21}$$

3.5.2. Preprocessing of Data

The data serving as basis for the proposed computation of recommendations has to be pre-processed in order to provide highly suitable recommendations and reaching the best possible results for the users of the system. These pre-processing steps include the following tasks:

- Stripping the data set from all tweets not containing any hashtags as these do not contribute to the recommendation process and would unnecessarily blow up the fulltext index and hence, slow down the recommendation computation process.
- Removal of all retweets as these would lead to recommendations based on the originally retweeted messages (if it is contained in our corpus) and would hence distort the evaluation results.
- Removal of all tweets containing more than 6 hashtags from the data set. Such tweets are mostly regarded as spam tweets and hence, may decrease the quality of the resulting hashtag recommendations. This is especially the case if trend-stuffing (cf. Section 3.2.4), as such tweets contain various hashtags which are not related to the tweet's content and also not related to the other hashtags used within the tweet.
- Removal of tweets which only consisted of hashtags as there is no information left from which the algorithm could compute recommendations from.
- Computation of statistics required for ranking computations later in the workflow as e.g., the overall popularity of a certain hashtags (i.e., to count and store the overall occurrences of each single hashtag within the data set).

3.5.3. Evaluation Algorithm

The evaluation of the proposed approach is done via a Leave-one-Out-Test [21]. Such a Leave-one-Out-Test is a traditional evaluation method for the assessment of the quality of recommendations. Basically, such an experiment is based on partitioning the test data (in our case the set of tweets contained in the data set described in Section 3.2). In the case of Leave-one-Out, one single item of the data set is withheld and constitutes the test item whereas the remaining entries are used for the computation of the recommendations. Subsequently, the recommendations are compared to the withheld item in order to evaluate the computed recommendations. Our actual evaluation algorithm is shown in Algorithm 23.

1. Randomly select a tweet from the database.
2. Remove all hashtags from this tweet.
3. Use the tweet as input for the recommendation engine.
4. Call the recommendation process.
5. Compare the set of recommended hashtags to the set of hashtags used in the original tweet.
6. Compute the evaluation metrics recall, precision and F1-measure as described in Section 3.5.1.

The evaluation was done on a CentOS 5 system with 96 GB RAM and 2 quadcore processors. We chose to use a sample size of 1.000, i.e. for each evaluation run, 1.000 random tweets are taken and the evaluation procedure for these tweets is carried out. It is important to note that within one set of evaluation runs (i.e. for the different ranking methods and the different configurations), always the same set of random samples was used in order to guarantee reproducible results. As for the number of search results used for the initial setup of hashtag recommendation candidates (variable n within Algorithm 1), we chose to perform the evaluations with a value $n = 200$ as our experiments showed that incorporating a higher number of search results did not improve the evaluation results. Besides, limiting the number of search results also cuts down the computation time for the evaluations.

As for the content of the tweets, we used standard stopword-lists (as provided by the Lucene engine[16]) in order to filter out (English) stopwords.

[16]Lucene's set of stopwords for the English language is: {a, an, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, such, that, the, their, then, there, these, they, this, to, was, will, with}.

```
Data: Set T of all tweets within the data set
Result: Evaluation of Recommendation Algorithm
begin
    // Initialisation
    randomTweet, inputText := null
    hashtagRecommendations, evaluationResults, hashtags := { }
    numberOfCorrectRecommendations := 0
    // Get random tweet from T
    randomTweet = getRandomTweet(T)
    hashtags = extractHashtags(randomTweet)
    inputText = removeHashtags(randomTweet)
    // Get recommendations
    hashtagRecommendations = getRecommendations(inputText)
    // Evaluate Recommended Hashtags
    foreach r within hashtagRecommendations do
        if $r \in$ hashtags then
            numberOfCorrectRecommendations++
        end
    end
    evaluationResult = computeMetrics(
      inputText,
      hashtagRecommendations,
      numberOfCorrectRecommendations)
    return evaluationResult
end
```

Algorithm 2: Basic Evaluation Algorithm

We varied the configurations of the evaluation runs in regards to the following features:

- Similarity measure
- Weighting scheme
- Ranking algorithm of hashtag recommendation candidates
- Various options for the similarity measures (e.g., the size of n for nGrams)

- Percentage of the already entered message in order to evaluate how the quality of recommendations improves in the course of entering a tweet, i.e. the refinement of recommendations
- Number of recommended hashtags (i.e. we varied k for the top-k recommendations)

These different configurations and the according evaluations are explained and discussed in detail in the following sections[17].

3.5.4. Results

In the following section, the results of the previously described evaluation are presented. Firstly, the performance of the different similarity measures is presented and discussed and subsequently, the results obtained by the proposed ranking methods are elaborated on.

The set-based Dice and Jaccard coefficients inherently perform equally as these two measures are monotonically related, i.e., the order of hashtags resulting from Dice and Jaccard are always equal. Therefore, we chose to only incorporate the Jaccard coefficient in our evaluations throughout the remainder of this chapter.

Similarity Measures

In order to be able to estimate the performance of the proposed similarity measures, we performed the experiments using the scoreRank ranking method which ranks the hashtag recommendation candidates solely based on the similarity of the input tweet and the tweet containing the hashtag recommendation candidates (in the case of a hashtag stemming from multiple tweets, the highest similarity value is used). Hence, the suitability of the different proposed similarity measures can be evaluated directly.

Figure 3.8 features a plot of the recall@k values obtained by hashtag recommendations computed based on the proposed similarity measures using the scoreRank ranking method. This plot shows the performance for the top-k recommendations in terms of the recall values on the y-axis for top-k recommendations, where k is set to 1, 3, 5, 10, 15 and 20. This plot clearly shows that the vector-based cosine similarity measures outperform the set-based and the lexical similarity measures. Cosine similarity with the BM25 weighting scheme is able to achieve a recall value of 13.49% for $k = 1$. With increasing k, also

[17]As not all the evaluations are discussed here in detail, the complete set of charts can be found in Section A.2 of the Appendix.

the recall value rises monotonously, achieving a recall value of 29.16% for 20 recommendations. Performing a term frequency-inverse document frequency (tf-idf) weighting results in a recall value of 29.48% for 20 recommendations. However, for k being set to the range of 3 to 15, term frequency-inverse document frequency weighting performs better. Another important finding of this evaluation is that the lexical Levenshtein distance measure performs better than the Jaccard coefficient. For k being set to 1, these two measures achieve recall values of 2.74%. With $k = 20$, the recall value is 7.92% for both Dice and Jaccard coefficients. The better performance of the Levenshtein distance can be lead back to the fact that this measure is more fine grained than the bag-based measures where two words are not considered to be equal if these are not exactly matching. Levenshtein is able to cope with such small distances of terms or tweets and hence, is less rigid.

The good performance of the two vector-based similarity measure regardless of which weighting schema is used can be explained by the weighting of terms which allows for a lowering of the influence of terms which occur frequently. These terms are presumably not relevant or characteristic for tweets and hence, these terms do not have any decisive power for the similarity computation. In contrast, terms which occur infrequently across the corpus are crucial as the allow for a strong characterization of tweets and hence are a decisive factor for the similarity. Two tweets which both contain a very sparsely used term are most likely to have similar content whereas two tweets containing a frequently used term may not be related at all in regards to content.

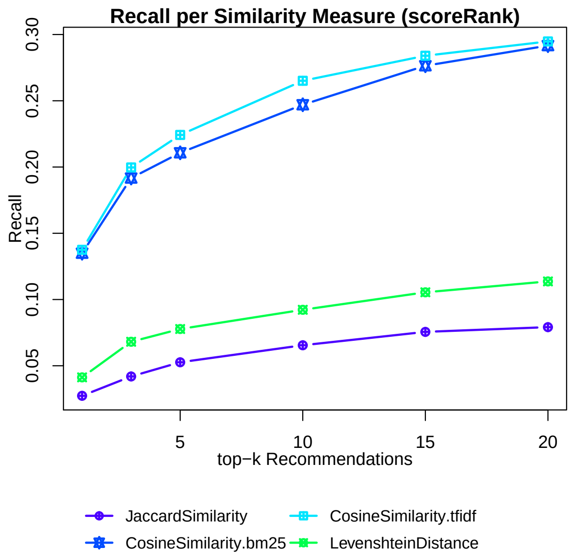


Figure 3.8.: Recall for Top-k Recommendations for Different Similarity Measures (Ranking Strategy: scoreRank)

Figure 3.9 depicts the precision values obtained by the various similarity measures (again, scoreRank was used for the ranking of the recommendation candidates). As can be seen, the vector based cosine similarity measure perform best, followed by Levenshtein distance and the bag-based similarity measures. The best performing similarity measure, cosine similarity with term frequency-inverse document frequency weighting obtained a precision value of 16.90% for $k = 1$. However, already at $k = 5$, the precision values drop to 7.45% for tf-idf weighting and 7.08% for BM25 weighting, reaching a low of 3.87%, resp. 3.74% for 20 recommendations. These low precision values can be explained by the observation that tweets contained in the underlying data set on average contain 1.32 hashtags (for further statistics see Section 3.2). Hence, already when recommending five hashtags, on average only one of these hashtags was originally used and hence, at most one hashtag can successfully be recommended. This naturally leads to low precision values as—in such a case—four of the five recommended hashtags did not match the originally used hashtags. Hence, precision values drop significantly with $k > 2$.

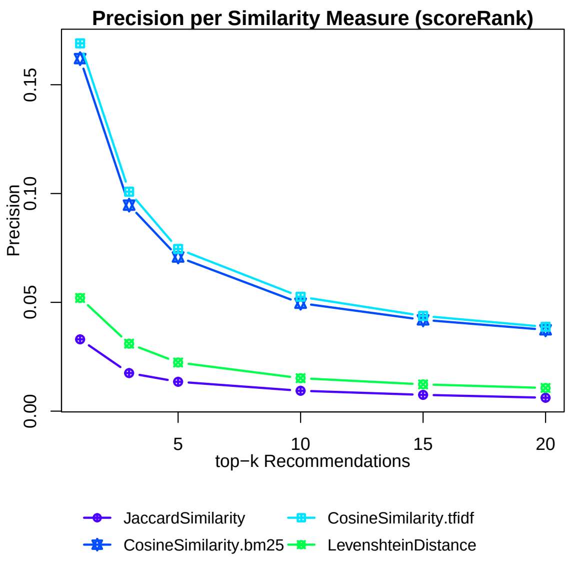


Figure 3.9.: Precision for Top-k Recommendations for Different Similarity Measures (Ranking Strategy: scoreRank)

Figure 3.10 shows the F1-measure for the different similarity measures based on the top-k (k being set to 1, 3, 5, 10, 15 and 20). This once again shows the above described findings where cosine similarity performs best, followed by Levenshtein and Dice and Jaccard coefficients.

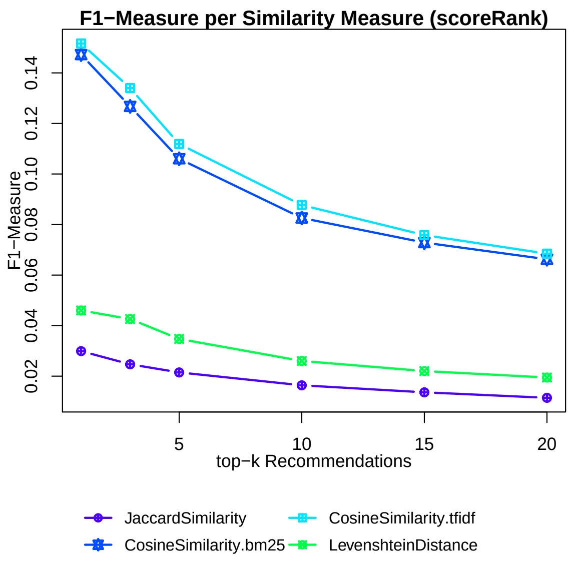


Figure 3.10.: F1-Measure for Top-k Recommendations for Different Similarity Measures (Ranking Strategy: scoreRank)

Ranking Strategies

As for the evaluation of the different ranking mechanisms, we performed an evaluation based on all available similarity measures. In particular, we firstly performed a recall@k evaluation. Figure 3.11 contains the recall values for all possible ranking mechanisms for all proposed similarity measures for the top-k recommendations (k being set to 1, 3, 5, 10, 15 and 20). It is important to note that the number k of recommended hashtags is most likely to be rather low in a hashtag recommendation application. This is due to the fact that the cognition of the user and also the space available for displaying the recommended hashtags is rather limited. As the process of entering a tweet is traditionally performed in a short period of time, a user cannot be presented with a huge list of recommendation candidates. The user is rather presented with a short, easy to comprehend list of possible hashtags. Hence, the evaluation results for top-k recommendation with the value k being set to values $k < 10$ is very important and once more strengthens the importance of a highly suitable ranking strategy.

Our evaluations showed that recCountRank and also—regardless of which ranking method was used—cosine similarity with both BM25 and term frequency-inverse document frequency perform by far better than the other proposed ranking methods. Both weighting schemes for cosine similarity perform almost equally for all ranking methods.

Overall, the best results in terms of recall were accomplished by the recCount--Rank strategy. This ranking strategy achieves a recall value of 30.36% for $k = 20$ and is based on the number of tweets within the set of the 200 most similar tweets from which a certain hashtag recommendation candidate stems from. Hence, the more similar tweets contain a certain hashtag, the higher is the ranking of the hashtag. However, the scoreRank strategy, where the similarity score between the input tweet and the respective tweet holding the hashtag recommendation candidate is directly used for the ranking of hashtags performs equally. Using scoreRank, a recall value of 29.48% for 20 recommendations with term frequency-inverse document frequency weighting. For BM25 weighting, a recall 29.16% was accomplished. The recCountRanking strategy is performed on a broad basis of tweets in a (democratic) voting process. The more similar tweets feature a certain hashtag, the higher is the ranking of the corresponding hashtag. In contrast, the scoreRank strategy is based on the most similar tweet from which the hashtag recommendation candidates are extracted from. Both of these methods showed to be suitable for the recommendation of hashtags.

The globalPopularityRank strategy achieves a recall value of 21.19% for $k = 20$ and 12.43% for $k = 5$ (cosine similarity with BM25 weighting). Considering the fact that regardless of the content of the tweet, the globally most popular hashtags within the data set are recommended, these results can be considered as a baseline for the evaluation recommendation. The recall value of more than 21% for this rather simple ranking strategy can be lead back to the distribution of hashtag popularity (as described in Section 3.2.3 and in Figure 3.2). As many of the tweets featured in the data set (and hence also featured in the set of random tweets which we used as input for the evaluation) contain these popular hashtags, recommending the most popular hashtags is bound to match some of the originally used hashtags. However, our approach is intended to provide content-sensitive recommendations and hence, globalPopularityRank is rather seen as a method suitable for e.g. recommending hashtags for input tweets for which no similar tweets are contained in the data set and hence, other ranking methods are not able to provide recommendations.

As for the date-based ranking strategies, the recall values obtained are expectedly low (as can be seen in Figure 3.11(d) and Figure 3.11(e)). The computed recall values for $k = 5$ are 3.40% for dateAvgUsageRank and 3.93% for dateRecentUsageRank. For $k = 20$, these two ranking methods achieve recall values of 7.21% and 8.11%, respectively. This is due to the fact that no content information is incorporated in the ranking. However, the proposed date-based ranking strategies are not aimed at being stand-alone ranking methods, they are rather aimed at being able to incorporate timely aspects into hybrid ranking strategies. The evaluation for hybrid ranking strategies can be seen in Section 3.5.4.

The evaluation of ranking methods showed recCountRank and the scoreRank methods performed best. Also, these evaluation result show that the similarity of messages from which hashtags are stemming from can be a good indicator for how likely it is that a hashtag is suitable for a given tweet.

Refinement of Recommendations

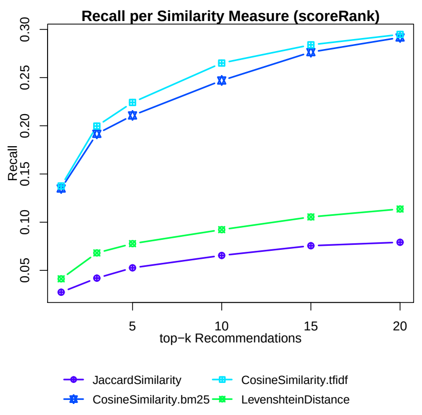
This evaluation is aimed at showing how the quality of recommendations improves as a larger and larger portion of the tweet is entered, i.e. as the user enters the input tweet. While the user is typing, the hashtag recommendation computation is performed on the fly. With every newly added character, the recommendations are recomputed and get refined. The charts shown in Figure 3.12 depict the recall values of the given recommendations (top-k recommendations, where k is set to 20 recommendations) where the x-axis features the percentage of the original tweet entered serving as input for the recommendation computation process (25, 50, 75 and 100% of the original tweet). I.e. for an original tweet featuring 100 characters, we used the first 25, 50, 75 and 100 characters and performed the hashtag recommendation process for each of these (new) input tweets aiming at evaluating the behaviour of recommendations during the process of entering a tweet.

Figure 3.12 shows the recall values for this evaluation for all the given ranking strategies and similarity measures. This figure shows that recCountRank and also scoreRank perform best also for a small number of characters entered when using cosine similarity. These ranking strategies reach recall values of 22.02% (recCountRank), 20.10% (scoreRank) for input tweets for which 50% of the original tweet were entered. Considering the fact that 65.79% of all hashtag usages occur after 30 characters (as shown in Section 3.2.5), these results underline the suitability of the proposed ranking methods.

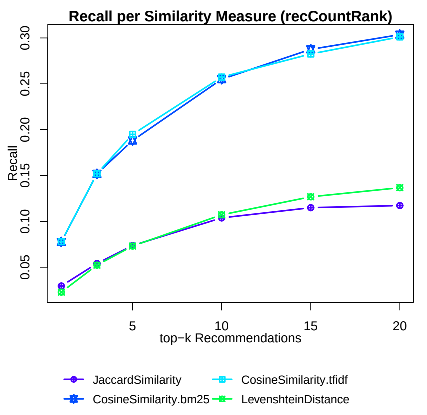
Further evaluation results concerning the precision and the F1 value in terms of refinement of recommendations can be found in Section A.2.4 and A.2.5.

Hybrid Ranking Strategies

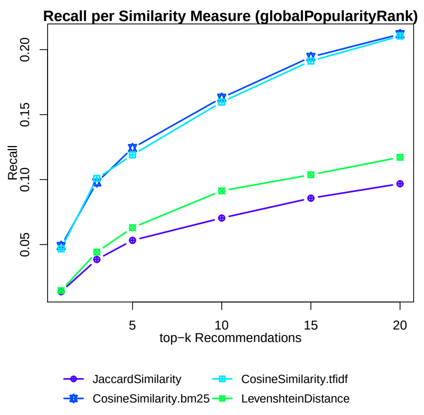
As proposed in Section 3.5.4, also hybrid combinations of the proposed ranking strategies are evaluated. We chose to evaluate all possible combinations of ranking strategies with a weighting factor α which is set to 0, 0.2, 0.4, 0.6, 0.8 and 1.0 in the evaluations. The hybrid ranking strategies combining ranking methods A and B are referred to as "rankingMethodA.rankingMethodB" in the following. E.g., for "scoreRank.globalPopularityRank" a weighting factor of $\alpha = 0$ means that in this particular configuration, only scoreRank is used whereas a weighting factor of $\alpha = 1$ means that only globalPopularityRank



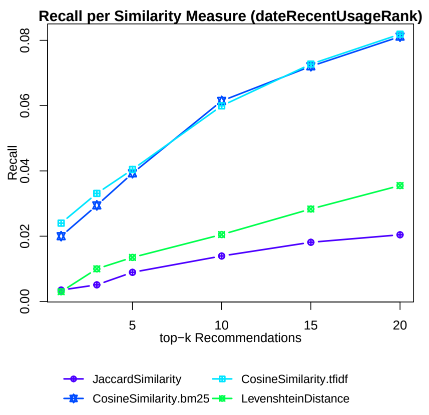
(a) scoreRank



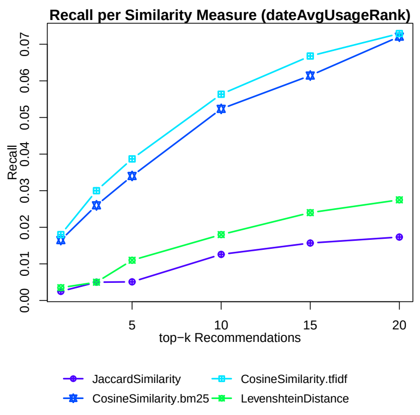
(b) recCountRank



(c) globalPopularityRank

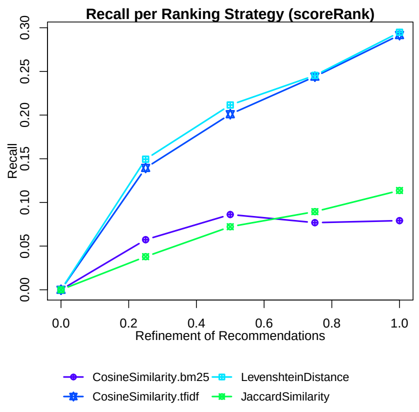


(d) dateRecentUsageRank

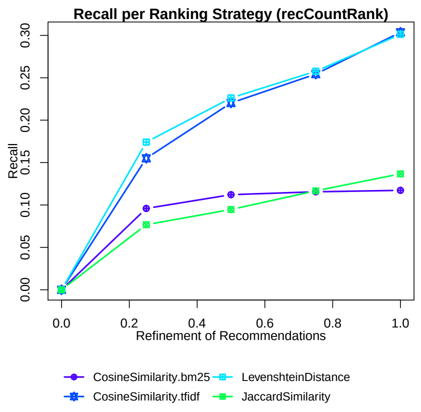


(e) dateAvgUsageRank

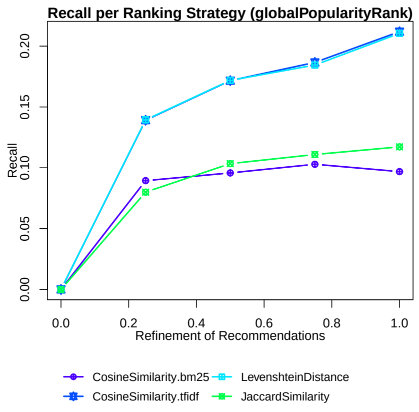
Figure 3.11.: Recall Values for Top-k Recommendations



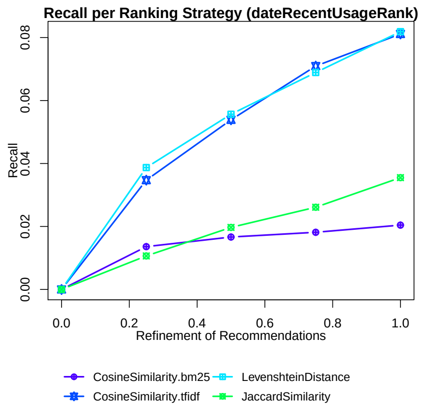
(a) scoreRank



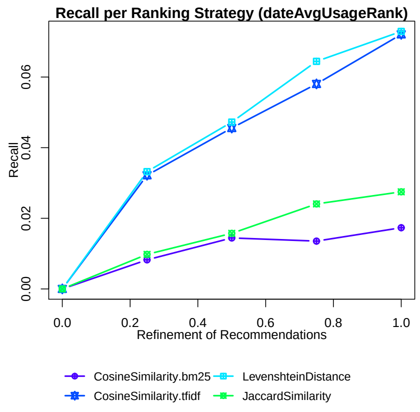
(b) recCountRank



(c) globalPopularityRank



(d) dateRecentUsageRank



(e) dateAvgUsageRank

Figure 3.12.: Recall Values for all Ranking Strategies with Increasing Percentage of Tweet Entered

is used. For all other values α, both of the ranking methods are used and combined according to the formula presented in Section 3.5.4.

Figure 3.13 shows the recall values of hybrid rankings (based on cosine similarity and BM25 weighting and top-20 recommendations). This evaluation shows that hybrid ranking strategies which incorporate either recCountRank or scoreRank perform best. The best achieved recall value is 32.77% for a combination of scoreRank and recCountRank with a weighting factor $\alpha = 0.6$. Naturally, the combination of the two best performing ranking methods also leads to the best results. Still, the combination of these two ranking methods lead to an improvement of the single ranking methods (2.41% improvement compared with recCountRank and 3.61% improvement for scoreRank). Another finding of this evaluation is that the temporal ranking methods (dateRecentUsageRank and dateAverageUsageRank) do not improve the recall values when used in a hybrid rank in combination with scoreRank or recCountRank at any given weighting factor. Furthermore, both the bybrid ranking methods recCount.globalPopularity and scoreRank.globalPopularity perform very stable for $\alpha \geq 0.2$

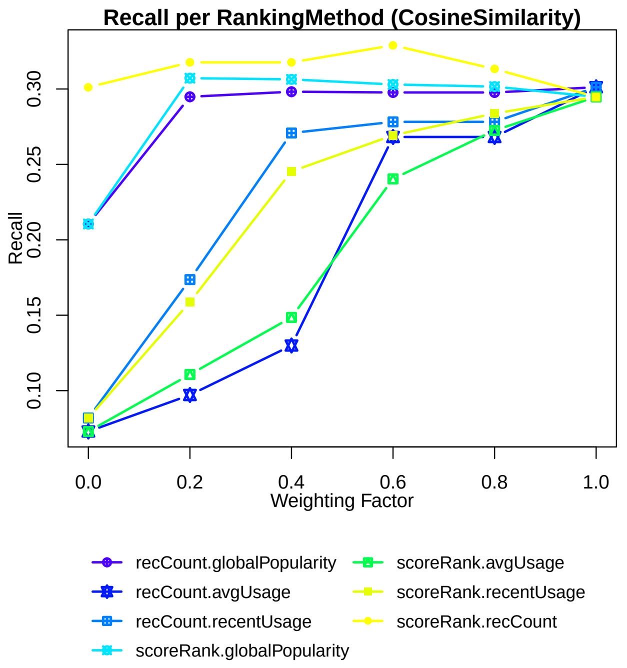


Figure 3.13.: Recall for Hybrid Ranking Strategies (Similarity Measure: Cosine Similarity w/ BM25 weighting, Top-20 Recommendations)

3.6. Principal Architecture and Implementation

The following section describes the system architecture and the main components of the prototype underlying the evaluation featured in this thesis. Subsequently, the architecture of a highly scalable system enabling live recommendations on huge amounts of twitter data is introduced.

3.6.1. Prototypical Architecture

The prototype was implemented in Java aiming at an efficient prototype enabling conducting the proposed evaluations which also was flexible enough to easily change parameters, similarity measures, etc. A component diagram can be seen in Figure 3.14, the components are described in the following.

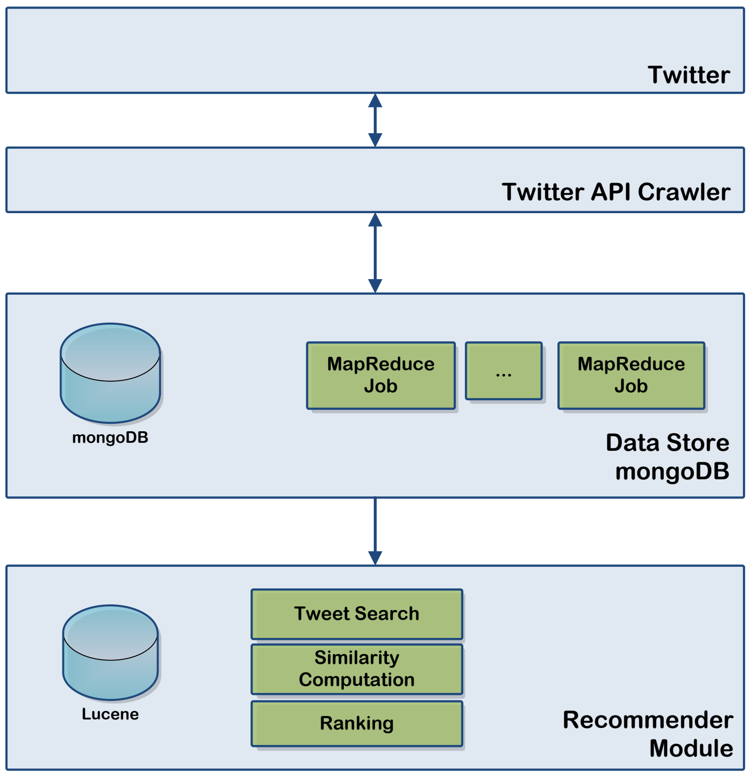


Figure 3.14.: Prototype Component Diagram

Crawler The crawling of tweets is the basis for the creation of a comprehensive data set which serves as the basis for the computation of recommendations. Hence, it is important to design a robust and flexible crawler. The implemented crawler (using PHP) uses the Twitter Streaming API[18], which provides access to a random sample of 1% of all tweets published on Twitter. The crawler sends requests to this API and stores the returned JSON stream containing the tweets and all related metadata to files. The tweets contained in those files are loaded into the data store periodically.

Data Store All crawled tweets and the related metadata are stored in MongoDB[19], a NoSQL database which provides MapReduce facilities [24]. MapReduce provides a computing model for a massively distributed computation based on key-value structured data. MapReduce works in two stages: (i) the *Map* stage where the task is split into subtasks and each node performs the according computations (mostly done on local data) and (ii) the *Reduce* stage where the results computed by the single nodes are collected and aggregated in order to obtain the final result.

For this prototype, statistics required for the computation of recommendations (as described in Section 3.2, like the number of total occurrences of hashtags, the average age of usage, etc.) are computed by MapReduce jobs.

Fulltext Index As for the fulltext index which holds all tweets and serves as the central component of the prototype, we rely on Lucene[20] which is a very efficient and popular open source fulltext index maintained by the Apache software foundation. However, Lucene did not provide implementations for the similarity measures we intended to evaluated. Hence, we implemented these measures for the Lucene store such that the used similarity measures can easily be interchanged and new measures can be implemented and added efficiently.

Recommender Module As for the recommender system, the core of the implementation was done in Java and relies on both the MongoDB storage and also the fulltext index. Furthermore, for the computation of the statistics underlying the ranking mechanisms for the recommendation candidates, MapReduce jobs were used.

Figure 3.15 depicts the (data) workflow underlying the prototype, consisting of a preprocessing workflow where data is crawled, stored locally in files and subsequently loaded into the MongoDB data store where the according

[18]`https://dev.twitter.com/docs/streaming-api/methods`

[19]`http://www.mongodb.org`

[20]`http://www.lucene.apache.org`

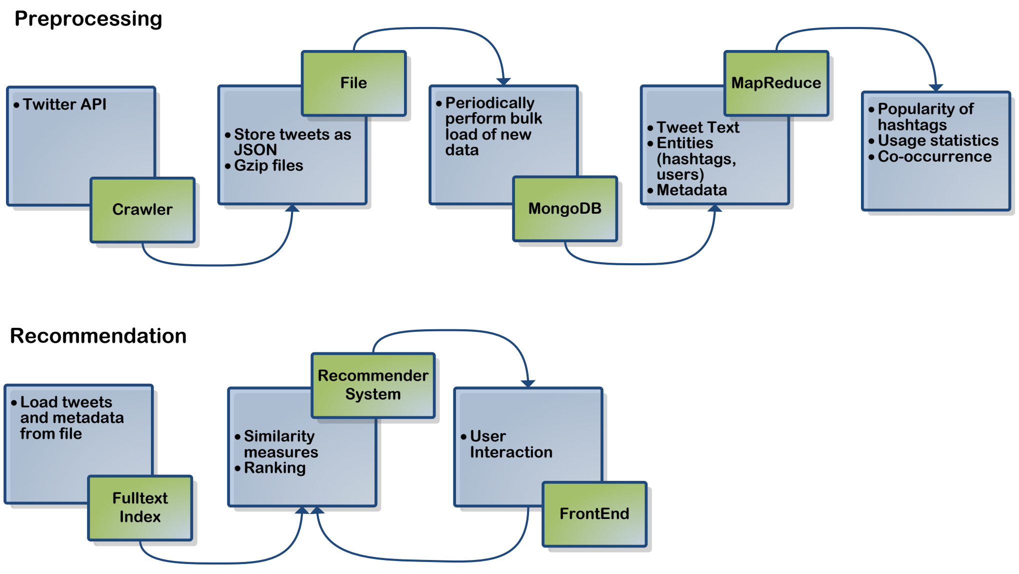


Figure 3.15.: Prototype Workflow

statistics are computed by performing MapReduce jobs. The recommendation workflow is based on the previously computed statistics and consists of loading the crawled data into the fulltext index, based on which the corpus is searched for similar tweets and the according recommendations are computed. The frontend is responsible for any user interaction and with every keystroke, new input data is delivered to the recommender system.

3.6.2. Architecture of a Scalable Live-System

Being able to perform live recommendations for a huge amount of data is a challenging task as it makes high demands on the underlying architecture. These requirements include being able to crawl millions of tweets per day, to store these and most importantly, to perform the recommendation computations and the according preprocessing steps based on hundreds of millions of tweets within the corpus in an efficient manner such that a multitude users can be provided with on-the-fly computed hashtag recommendations at the same time. In the following, we present an architecture for such a scalable and efficient live-recommender system for hashtags.

We propose to facilitate a highly distributed architecture underlying the actual recommender system. Hence, this architecture provides possibilities to scale out by adding further nodes. A component diagram can be seen in Figure 3.16. The different components are described in the following.

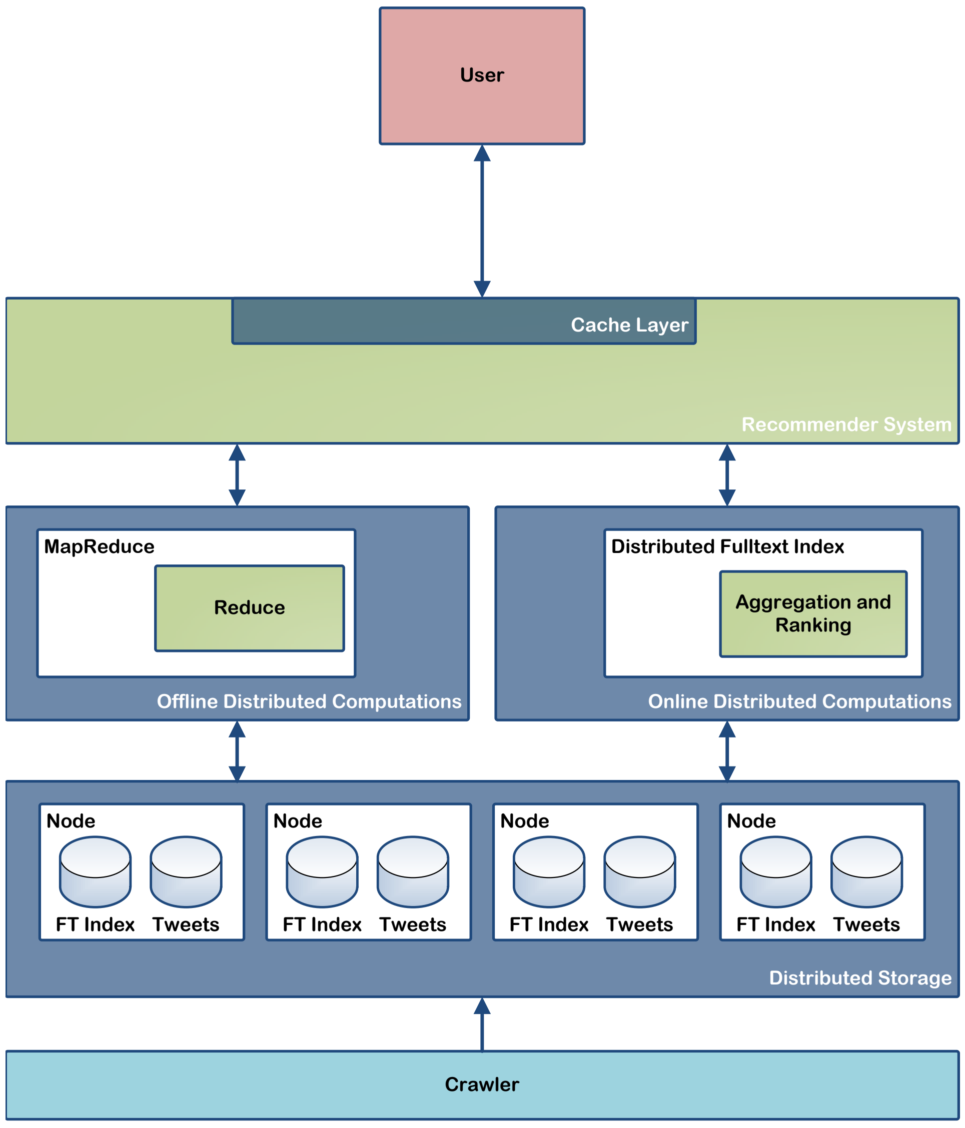


Figure 3.16.: Scalable Architecture

Crawler

As for the gathering of tweets underlying the recommender engine, a crawler is aimed at gathering a diverse and representative set of tweets in an efficient and robust manner. The current Twitter API freely provides a random sample of tweets offering a wide range of tweets in regards to countries, languages, topics, etc. Due to the current limitations in regards to the number of API calls allowed per hour, the set of crawled tweets could be restrained around certain languages or countries where the tweets originate from in order to focus the crawled tweets and to increase the fraction of valuable tweets in regards to their suitability for the later computation of recommendations.

Data Storage

The central storage facility is required to hold terabytes of data and to provide scalable, parallelizable access for any further computations. We propose to make use of a distributed file system which allows for data storage distributed among a large number of nodes. Such a distribution enables a redundant storage of the crawled corpus ensuring high availability of data. Based on such a storage system, a fulltext index can be shared among the nodes such that large amounts of indices may be stored and queried efficiently. Still, the crawler has to be able to cope with high amounts of incoming tweets.

Distributed Recommendation Computation

On top of the proposed distributed data storage, a framework for distributed computations is also required. This framework is required for two tasks: (i) the computation of statistics and heuristics underlying the recommendation computation and (ii) searching the sharded fulltext index. Hence, we propose to make use of MapReduce jobs based on the distributed file system for the computation of statistics and heuristics. These computations cannot be performed live due to time limitations and hence, are performed offline at regular intervals in order to keep the statistics up-to-date. The results are subsequently stored in the distributed storage system. As for the computation of search results based on the sharded fulltext index, after a distributed search of the indices on each single node, a means for the aggregation and ranking of the individual search results has to be implemented on top of the distributed file system. This allows for online fulltext search which serves as the basis for the live (online) recommender system, which also accesses the precomputed statistics.

In order to further enhance the performance of the system, we propose to introduce a cache layer to the recommender system component. This cache layer aims at identifying tweets possibly related to currently trending hashtags (thus, occurring at a high frequency) by precomputed patterns and keywords. These patterns are extracted and computed offline periodically by MapReduce jobs. If such a pattern matches the tweet the user is currently entering, the hashtags associated with this pattern are presented to the user. If none of the patterns is matched, the standard recommendation computation is initiated. Again, the frontend is responsible for any interaction with the user and enables the user to enter tweets quickly and easily. The frontend presents the user with recommendations and with every keystroke, the current tweet content is sent to the Cache Layer and—depending on whether the cache layer contains any suitable recommendations or not—the recommender system in order to retrieve new and up-to-date recommendations suitable for the entered information.

As for updating the statistics and the sharded fulltext indices, the interval between updates is performance-critical as trends in regards to hashtag popularity may emerge at a high pace in Twitter (e.g., in the case of events and disasters, etc.). Hence, the goal is to find a balance between not being forced to constantly compute statistics and still being able to reflect trends for the recommendation of hashtags. Hence, a short computation time for statistics is required. This can be achieved by making use of incremental MapReduce jobs which compute heuristics for the newly inserted data and add these results to the already computed persistent results of previous MapReduce computations.

Figure 3.17 depicts the new workflow for the computation of recommendations.

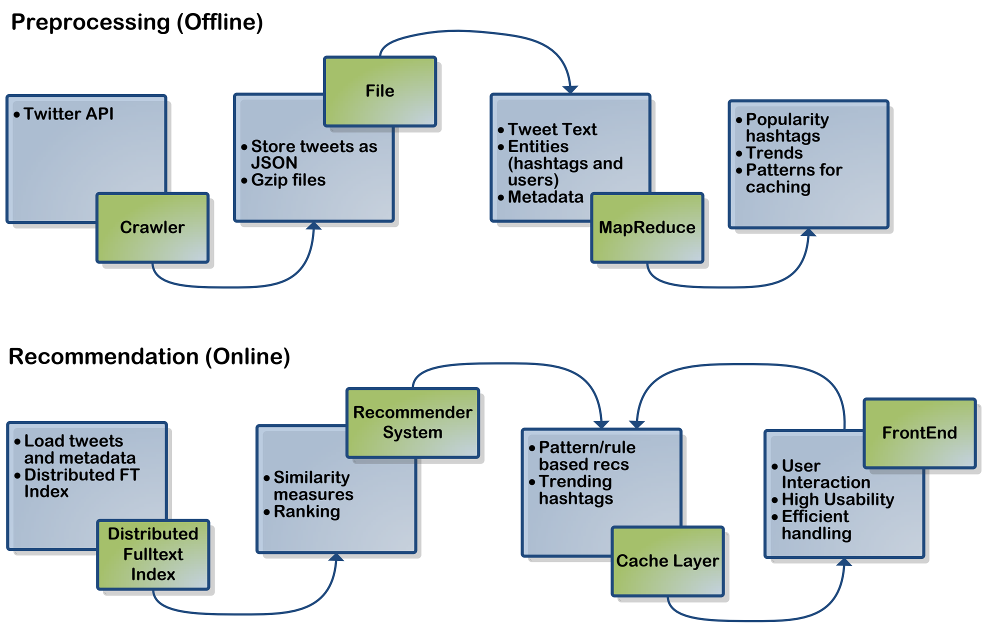


Figure 3.17.: Workflow for Live-Recommendations

3.7. Related Work

This section is concerned with research related to hashtag recommendations. Therefore, multiple areas of research are examined: (i) general findings and about the Twitter platform, its users and their behaviour, (ii) works particularly concerned with hashtags, (iii) recommendation-based approaches on the Twitter platform, (iv) the related research area of tag recommendations within various other social networks.

Our approach was introduced in 2011 [119] and is the first approach which deals with the heterogeneity and sparseness of hashtags within messages on Twitter by facilitating a recommender system for the suggestion of suitable and uniform hashtags. Kywe *et al.* [62] presented an approach which combines both the hashtags contained in the set of the most similar tweets and a user-profile which is based on the hashtags the user previously made use of (like we sketch in future work, cf. Section 3.8) in 2012. The data set underlying the evaluation of this approach contains 2.3 million tweets and was crawled by retrieving the tweets of 150,000 Singapore-based users. The authors evaluated the proposed approach in regards to the hit ratio based on historic data. They define the hit ratio as the number of evaluated tweets for which at least one hashtag was recommended correctly. Kywe *et al.* reached the best hit ratio of 37% with a configuration where 10 hashtags are recommended and the top-50 similar tweets and the top-5 similar users are incorporated into the recommendation process. However, if a tweet originally contains two hashtags, the hit ratio would be 1 whereas in our evaluation, the recall value would be 0.5. This difference in the evaluations makes a direct comparison of the performance of these two approaches hardly possible.

3.7.1. Twitter

Twitter has been a popular topic for researchers coming from very diverse disciplines. Due to Twitter's hugely dynamic nature, tweets have e.g. been facilitated to detect earthquakes or natural hazards in real time [92, 110]. Similarly, based on the dynamic nature and the huge potential stemming from millions of users making their opinion on certain topics public via tweets, researchers proposed to use such data to predict election results [109] or to predict stock market movements [12].

TwitterRank [113] shows that the follower-followee connection is based on homophily [71] which basically means that both users share interest in one or more particular topics. Based on a data set of tweets, the authors analysed the behaviour of users when following back their own followers. They found that the number of followees of a user is directly related to the number of followers, the more followees a user has, the more followers she has (and vice-versa).

In regards to the following-back behaviour of users, the authors showed that 72.4% of all Twitter users follow back more than 80% of their own followers. Moreover, 80.5% of all users have 80% of their followees follow them back. In order to be able to compute shared topical interests, the authors detected the topic of tweets by making use of a Latent Dirichlet Allocation Model. Based on this information, the authors propose to define the influence of a user not solely by the number of followers a user has, but also on the expertise of Twitter users in different topics. Hence, the authors propose new measure the influence of Twitter users named TwitterRank. This measure is based on the PageRank algorithm [79] originally published by Brin and Page which was created to compute the influence of websites based on the link structure of the web. PageRank features a random surfer model which computes the probability of a surfer who randomly follows links within webpages to come across a certain website. In order to adapt this model for measuring the influence of Twitter users, the random surfer model has to be adapted such that the following of links corresponds to traversing the the follower-followee graph of Twitter users. Furthermore, the random surfer model is adapted such that the following of links is based on the topical relatedness of the according users.

The work by Huberman, Romero and Wu [49] was one of the first works dedicated to an analysis of the Twitter network. The main finding of the analysis the authors conducted based on a set of 309,750 Twitter users is that the number of followers of a user is directly related to the number of posts the user publishes. Thus, the more followers a user has and the more attention a user receives by her followers, the more posts are published by this specific user. However, this hypothesis only holds up to a certain number of followers, the number of posts eventually saturates as the number of followers still increases. They also found that there exists a relation between the number of friends (in this case a friend of a certain user A is defined as A having sent more than two direct messages to another user B). This function of the number of posts and the number of friends of a user does not saturate and hence the authors propose that the number of friends of a user is a more accurate estimate for the activity of a user within the Twitter network. Another finding of this work was that the number of friends is very small in comparison to the number of followees a user has. As the number of followees increases, the number of friends of an average Twitter remains constant after having reached a certain number of followees. The authors explain this by the fact that Twitter enables users to easily follow other users. However, friendship requires more effort and at least two direct messages exchanged.

In regards to search facilities on Twitter and the shortcomings of traditional keyword search, Abel *et al.* [1] propose a faceted search approach for tweets by enriching the semantics of tweets. This enrichment is achieved by extracting entities within tweets and linking these entities to external sources. Based on this information, facets can be created where the subject of the facet corresponds to the tweet, the facet type corresponds to the type of the entity (e.g., location) and the facet value corresponds to the mentioned entity (e.g., New York). This enables users to explore tweets by browsing through facets.

3.7.2. Hashtags

The topic of hashtags has become very popular for researchers throughout the last years. This section describes work concerned with hashtags and the hashtagging behaviour of Twitter users.

Tsur and Rappoport [108] propose an approach which aims at predicting the spread of an hashtags within the Twittersphere within a given time frame. The spread of hashtags (or ideas) is characterized by a normalized count of occurrences within the given timeframe. For the prediction, the authors base their approach on a regression model in which hashtags are presented by four different feature vectors: (i) the content of the hashtag (e.g., orthographic features, the length of the hashtag or the location of the hashtag within the tweets), (ii) global tweet features (the context of the hashtag characterized by the words appearing within tweets together with the hashtag), (iii) graph topology features (e.g., the retweet ratio or the average number of followers of users who used this specific hashtag) and (iv) global temporal features (the number of weeks since the first occurrence of the hashtag). The evaluations showed that a hybrid combination of the four proposed feature types yields the best results.

An approach widely related to the approach proposed in this dissertation is presented by Esparza *et al.* in [33]. The authors propose an approach aiming at automatically creating a categorization of tweets. A category can be characterized by a set of tweets containing information related to the according category. This approach characterizes a category by the set of terms contained in the tweets contained in the category. Hence, a bag-of-words approach based on the Lucene fulltext index (hence, tf-idf is used as a weighting schema and similarity measure) is proposed which, for each new input tweet, is compared to the already existent tweets and the related category. Subsequently, the top-ranked category is proposed to the user. This approach, in contrast to our proposed solution, adds another layer above the microblogs for their categorization and is only able to deal with a predefined set of categories. Furthermore, the initial assignment of tweets to categories has to be accomplished manually in a first step unlike our approach, which does not require such a manual preprocessing of the data set. Similarly, [55] aim at assigning tweets to six different topical categories (books, games, movies, photography, politics, sports) which are extracted from the hashtags occurring in these tweets. The categorization is based on feature vectors containing (i) the content of the original tweet without an actual URL, (ii) the content of the tweet including the URL, (iii) HTML of the webpage the URL links to and (iv) metadata extracted from this webpage. The authors then used a Naive Bayes classifier. The results showed that incorporating metadata extracted from (structured) webpages can contribute to a better categorization of tweets.

Potts *et al.* [85] studied the hashtagging behaviour of users during two disasters: the earthquake in New Zealand in 2010 and the earthquake in Japan in 2011. During these two quakes, Twitter was extensively used to spread information about the happenings. The authors analysed the hashtags used during these two disasters and found that a very heterogeneous set of hashtags was used. The authors showed that users tried to fight this heterogeneity manually, like in the following tweet.

```
#earthquake Apparently the official hashtag is
#eqnz; #doingitwrong. Glad someone's got priorities
sorted.
```

posted by @adzebill on 2010-09-04

The authors call for an improvement of the way how hashtags are used and treated pointing at the proliferation of hashtags during the above mentioned natural hazards, which is the main problem which is tackled in the approach presented in this thesis.

Lehmann *et al.* [65] analyzed the popularity of hashtags over time. They found that there are three temporal patterns regarding hashtag popularity: (i) continuous popularity, (ii) periodic popularity and (iii) isolated peaks. The latter pattern was analyzed in detail and the key finding of this work is that hashtags that are primarily used before reaching a peak in terms of popularity are typically related with either scheduled events or specific moments in time. Hashtags which are mostly used before and after the peak are mostly associated with endogeneous factors where attention is intensified by a propagation of the hashtag. Furthermore, hashtags which are primarily used after already having peaked are mostly related to unexpected events.

The behaviour of scientists on Twitter especially during conferences and as a medium for spreading scientific messages has been studied by Letierce *et al.* [66]. The authors first analysed how researchers of the semantic web community make use of different services to spread information. This study revealed that the most popular medium for researchers of this community is Twitter (92% of all participants do have a Twitter account). In a next step, the authors analysed tweets related to three different mayor conferences in the field. The mayor findings of this study in regards to hashtags are that (i) the research community members make use of hashtags more frequently aiming at increasing their network and (ii) hashtags mostly revolve around technical terms, events and fields of research.

Carter *et al.* [18] observed that hashtags are not only exclusively used for making tweets more searchable, but rather also serve as the basis for statistics about trending topics within the Twittersphere. The fact that a multitude of hashtags describe the same topic biases such statistics. The main goal of this work is the robust translation of hashtags between languages as the authors state that hashtags are used differently in different languages and in different locations. This translation is based on gathering a large set of messages containing the hashtag which has to be translated, translating these messages and extracting the most characteristic terms in regards to their tf-idf measure. These terms are then used for querying Twitter for messages containing these terms. From the resulting tweets, the hashtags are extracted and proposed as translations for the input hashtags.

Hepp [41] proposed to incorporate triples into microblogs aiming at creating a common knowledge representation. In his work, Hepp presents a syntax for the inclusion of triple-statements into microposts. By using such triples in microposts, users are enabled to define relations between different hashtags (e.g., subtag-of relations). Hence, an ontology of hashtags is created which can be exploited for the resolution of synonymous hashtags or simple reasoning tasks. However, the inclusion of triples is not popular on Twitter. Furthermore, by defining an ontology of hashtags and other entities, the problem of synonyms is not resolved, it is rather shifted towards a broader search task where searches may be conducted by using all hashtags declared as synonymous within the ontology in order to find all relevant information. This is—in contrast to our proposed approach, which aims at keeping the hashtag vocabulary homogeneous already at the time of insertion—more a treatment of symptoms of the proliferation of hashtag vocabularies. Another semantic web-based approach has been proposed in [81]. Passant *et al.* propose to incorporate microblogs and the according metadata into the Linked Open Data cloud by making use of the very popular ontologies FOAF (Friend-of-a-Friend) and an extension of SIOC (Semantically-Interlinked Online Communities). By facilitating semantic lookup services, hashtags are turned into unique URIs and are enriched with e.g. geographic information and due to this added information, the information contained in the microblog is also accessible by other applications and are openly (re)-usable. Stankovic, Rowe and Laublet [103] also propose to extract information from tweets and map this information with Linked Open Data sources, especially in the field of conference talks.

Efron [29] also depicts an approach aiming at retrieving hashtags for a certain topical query in order to find tags and hence conversations which might be interesting for a user to follow. However, the authors mainly propose such an approach for the expansion of queries for relevance feedback.

Laniado and Mika [63] present different metrics which aim at identifying hashtags which serve as strong identifiers for tweets, namely (i) frequency, (ii) specificity, (iii) consistency of usage and (iv) stability over time. The authors propose to make use of the frequency of the hashtag as either the number of users which make use of a certain hashtag or as the number of messages containing this hashtag. Beside the frequency, the specificity describes how similar the semantics of the hashtag and the word underlying the hashtag are. The consistency of usage is intended to ensure that a hashtag is used in the same context over time. The stability over time is used to make sure that a hashtag is used over a long period time and not subject to some temporally local trend.

Kouloumpis *et al.* [58] make use of hashtags to create a training set for sentiment analysis on tweets. They use the most frequent hashtags occurring in their data set, manually assign a sentiment (negative, positive or neutral) to these hashtags and use this data for the training of their classifier.

3.7.3. Recommendations in the Twittersphere

In the field of recommender systems, there have been numerous approaches aiming at facilitating Twitter for recommendations or to provide Twitter users with recommendations.

The Twittomender approach by John Hannon *et al.* [40] provides evaluations for different approaches for establishing a profile of a certain user. These profiles are used for both user search and—more importantly—recommending users which might be interesting to follow for a certain user. The authors propose to use content-based profiles and profiles based on collaborative filtering. Three content-based profiles are proposed: (i) the profile is formed by the tweets, (ii) the tweets of all followers of the user are used for the profile or (iii) the tweets of all followees are used. As for the CF-based profiles, the authors state two approaches: (i) the use of all followees of the user and (ii) the use of all followers of the user. Apparently, a hybrid approach combining both CF-based profiles performs best in terms of precision. A similar approach has been facilitated by [6], however in this approach, also the topology of the network surrounding the user is taken into account.

The Tadvise project [78] aims at finding topical hubs within Twitter users which can be used to broadcast tweets to a broader audience. The system is able to recommend (i) users that may act as hubs for broadcasting a message about a certain topic by retweeting it to their followers and (ii) a categorization of followers of a certain user in terms of identifying all users tagged with a term mentioned in the given input tweet (to find a relevant audience for the given tweet). Therefore, the authors propose to build a user profile for every participating user. Such a user profile is built by analyzing the Twitter lists the corresponding user is part of. As lists can be seen as a tag for a set of Twitter users, the authors make use of these lists and compute a weighted list profile based on the given information. The weighting is performed based on the ranks of the users which are part of the list, in particular these ranks are summed up for the weight. This ranking of users is computed based on the number of followers of the according user. As for the detection of users tagged with a certain term, the authors propose to build the aforementioned user profiles of the followers and the followers of the followers of the given user. These user profiles are subsequently aggregated and then clustered into those tags which frequently occur and those tags which occur infrequently. The former are then recommended to the user, as they are likely to be interested in the given tweet. As for the detection of hubs for a certain tweet, the authors propose to build a directed graph by adding all the followers of the given user. Moreover, all followers of the followers who were tagged with a term (in the computed user profile) mentioned in the input tweet are added to the graph. Then, the hubs are detected within the graph by selecting the users having the highest number of followers.

The work published by Chen *et al.* [19] presents a *content recommender system* named *zerozero88* which is based on Twitter. Its aim is to provide users with content recommendations which are interesting to the user. More specifically, URLs containing information the user might be interested in are recommended. The authors especially focused on three main modules: (i) the generation of candidate sets of tweets, (ii) the ranking of tweets based on topic relevance approaches and (iii) the ranking of tweets based on a social process. The generation of a set of candidate URLs based on which the recommendation of content for a certain user A is implemented by two different approaches: one candidate set is generated from all tweets of all followees and the followees of the followees of user A. The authors assume that a user A always follows other users based on his interests and therefore, URLs posted by such users seem to be highly interesting to user A. The second approach is to use the most popular tweets on Twitter, regardless of who posted the URL. As for the ranking of tweets based on topic relevance, the authors propose to use bag-of-words-profile to determine the relevance of a certain topic for user A. The so-called Self-Profile of a user is the bag-of-words of all the user's tweets and the so-called Followee-Profile is defined by the Self-profile of all followees of the user. Based on these two profiles, the ranking of URLs is computed by the overlap between the bag-of-words profile for a certain URL and the two types of profiles. The authors also propose to make use of social factors for the ranking. Hence they present a social ranking process which also incorporates the relation of user A to user B who posted a certain URL and the frequency of posts by user B. The evaluation of these approaches was conducted by user tests which showed that the best performing approach was the combination of the candidate set generated by the tweets of all followees and their followees and a ranking based on a social ranking process based on the Self-Profile of the user. For this combination of approaches, the test users attested that 73% of all URLs are of interest for them.

Twitter is also used to enhance ranking mechanisms of recommender systems. Phelan *et al.* [84] propose a recommender system which aims at recommending news items to users. Therefore, RSS feeds of news articles are crawled which serve as the basis for the recommender system. At the same time, tweets are crawled which are used to enhance the ranking of the recommended news articles. This is done by computing the overlap between the news articles and the tweets, i.e. the more words are mentioned in both the tweet and a corresponding article, the higher the rank of the article is. The authors propose three different ranking strategies: (i) ranking based on the public timeline (the most recent tweets on the Twitter platform), (ii) ranking based on the tweets of friends of the user the recommendations are computed for and (iii) as a baseline method, a content-based approach is facilitated, where ranking is solely done based on how often a certain topic is contained in the whole RSS data set. The user evaluations showed that the friend-based ranking approach performs best in terms of click-thru data, whereas the analysis of the user questionnaire showed that users would prefer the ranking based on public tweets. The task of news recommendations has also been tackled by Abel *et al.* [2]. The authors present a news recommendation service for Twitter users and explore different ways of creating a user-profile aiming at high-quality of personalized recommendations. Such a user-profile can be built on top of (i) the hashtags in the user's tweets, (ii) the entities recognized within the tweets and (iii) the topics featured in the user's tweets. The authors furthermore propose to enrich the given tweets with either entities recognized within the tweets or entities extracted from URLs which were featured within the tweets. The best results in regards to precision and recall of the computed recommendations were achieved by using a entities-based user-profile which was enriched by external news resources.

As for the recommendation of user-created groups in the Orkut social network, Spertus *et al.* [101] evaluated different similarity measures for this recommendation task. These similarity measures rely on the overlap of users being part of the different communities. The authors argue that community memberships are rich enough for serving as a basis for recommending suitable communities to a community as a whole. Hence, the recommendations computed are not computed on a per-user basis, rather on a per-community basis. The authors did a test-user study and evaluated the performance of six different similarity measures in terms of user acceptance of the recommendations. The outcome of this study was that the relatively simple L2-measure (cosine similarity of the binary vectors of group memberships, i.e. the vector contains an entry for each user, 1 if the user is member of the according community, 0 if she is not a member) performed best.

3.7.4. Tag Recommendations

The recommendation of tags for (mostly online media) resources has been a popular research area since the spread of the web 2.0 paradigm. Traditional tagging systems are concerned with the annotation of items aiming at being able to enhance search capabilities on these items. Popular tagging systems are concerned with photos (e.g., the Flickr platform[21]), bookmarks (e.g., the delicious platform[22]) or scientific publications (e.g., Bibsonomy[23]). The difference between tag recommender systems and traditional recommender systems (as presented in Chapter 2) is that traditional recommender systems are based on two dimensions: users and items based on which recommendations are computed. Tag recommender systems add another dimension, namely tags.

Ames and Naaman [5] analysed the different incentives for users to tag photos and the authors propose a two-dimensional classification describing user motivation to tag photos. The first dimension is sociality which describes for whom (self or others) the tags are intended to be of use for. The second dimension is function which describes to main aim of the tag. This classification leads to four types of motivations: (i) self/organization: users use tags for themselves in order to be able to (re)find the previously uploaded photos, (ii) self/communication: users provide contextual information about the photos (e.g., names of people pictured, etc.) for themselves, (iii) social/organization: users tag photos in order to enable other users to find these photos and (iv) social/communication: users provide contextual information for other users. The authors found that the primary motivation of users to tag photos is organization rather than the communication factor.

Sigurbjörnsson *et al.* [99] proposed a tag recommendation algorithm for the Flickr platform. This algorithm is based on the co-occurrence of tags on

[21]`http://www.flickr.com`

[22]`http://delicious.com`

[23]`http://www.bibsonomy.org`

images, i.e. the fact that two tags are used for the same image implies that these tags are related. Hence, it is limited the sense that recommendations can only be provided for partly tagged images (at least one tag has to be added to an image in order to receive further recommendations). In contrast, Jäschke *et al.* [52] model the task of tag recommendations as a graph problem. The graph consists of vertices for users, items and tags and edges for the corresponding relations. Tag recommendations are ranked by a PageRank-like algorithm which reflects that tags which are used for important items by important users may also be important. Other tag recommender systems e.g. rely on learning algorithms, as Lipczak in [69] where for the recommendation of tags in the Bibsonomy bibliographic database, further sources are exploited. In particular, the authors propose to extract tag candidates from the item's title, tags which have already been used by the user and also tags which have already been used for a certain item. The evaluations showed that recommendations based the user's previous tags perform best. Other approaches are based on collaborative filtering (Nakamoto *et al.* [77]) or k-Nearest neighbour detection (Gemmell *et al.* [37]).

Still, the recommendation of tags for certain online resources is quite different from the recommendation of hashtags within microblogging services. Firstly, the time of recommendation is different: traditional tag recommendations are performed in a second step after the particular item was added whereas hashtag recommendations have to be provided already during the time of insertion or sending. Secondly, traditional tags are considered as metadata whereas hashtags also directly serve as content of a tweet. Furthermore, our current approach is solely based on the content of the tweet, which is at most 140 characters long, whereas traditional tag recommender systems are mostly based on more information about the items to be tagged. Another difference is the vast dynamicity and flexibility of hashtags in Twitter as new and trending hashtags may evolve at a fast pace due to the fast pace in which messages are spread on Twitter.

3.8. Conclusions and Future Work

In this chapter, we proposed a recommender system for hashtags in microblogging environments and the according recommendation and ranking algorithms. The comprehensive evaluation showed that such a recommender system is capable of providing the users of a microblogging platform with suitable recommendations for hashtags reaching recall values of about 33%. These recommendations aim at establishing a homogeneous vocabulary of hashtags which contributes to better search performance. Also, a more homogeneous set of hashtags is able to channel conversations about a certain topic.

Still, the proposed approach may be enhanced in future work. Our analysis of the crawled data set showed that 12% of all hashtag usages occur at the very first position of the tweet. This fact constrains our approach in a sense that—if there is no content of the tweet available—the proposed recommendation approach is not able to provide recommendations. However, bootstrapping the recommender system with (i) hashtags the user previously used in her tweets and (ii) the most popular overall hashtags, which might even be personalized (e.g., using geographic aspects), is a possible solution to overcome this well-known cold-start problem for recommender systems.

The crawled corpus is of reasonable size, however—due to the Twitter API constraints—only a small fraction of the corpus stems from the recent past which restricts the ability to dynamically respond to trends in regards to the provided hashtag recommendations. Hence, crawling at a higher bandwidth in terms of the number of gathered near-realtime tweets would contribute to a better recommendation performance for hashtags having become trending in the more recent past.

In regards to the semantics of tweets, various aspects may contribute to an optimization of the proposed approaches. As described in Section 3.5.4, the performance of the proposed approach is constrained by the vocabulary miss-match problem (as e.g. described in [72]). Hence, a very important future work task is performing a content analysis of tweets in order to be able to grasp the topic of the tweet in order to be able to perform content-sensitive hashtag recommendations. Such information can e.g. be incorporated to our approach by implementing query expansion in regards to the detected topics in order to be able to retrieve tweets which might not be syntactically similar, but semantically (topically) similar. Probabilistic topic modelling [104] as latent semantic analysis or latent dirichlet allocation have already been facilitated for the analysis the content of tweets, e.g. in [47, 56, 86]. Another approach is to make use of entity recognition aiming at extracting entities (persons, locations, events, etc.) from the tweets and matching these. This can be implemented by e.g. making use of the OpenCalais API[24] which provides a service for entity recognition. Also, sentiment analysis for Twitter messages has been addressed by various researchers (as in [80, 58]) and may be used to recommend hashtags according to the "mood" of the tweet or the user. This would also require a classification of hashtags into different sentiment classes. Yet, not only the sentiment of a tweet is relevant.

Concerning the ranking of hashtags, also metadata such as the geographic location of the user may be useful as hashtags can be ranked according to local trends and preferences. Furthermore, an analysis of the social network

[24]`http://www.opencalais.com/`

of a certain user is also part of future work as studies showed that a user is more likely to adopt hashtags which he is exposed to multiple times. Hence, as a user is exposed to the tweets of users featured in the user's social graph more often, these tweets and hashtags influence the tweeting behaviour of the respective user and hence, are worth incorporating in the hashtag recommendation approach.

In regards to the evaluation of the proposed approach, a user study may also contribute to better understand the user's needs and requirements. Furthermore, the conducted evaluation poses the problem that it is a rather narrow evaluation as it assesses whether the original hashtag of a tweet may be recommended. However, there may have been even more suitable hashtags which cannot be evaluated and assessed automatically. Such an evaluation can only be conducted by test-users in live experiments.

CHAPTER 4

Semistructured Information Systems

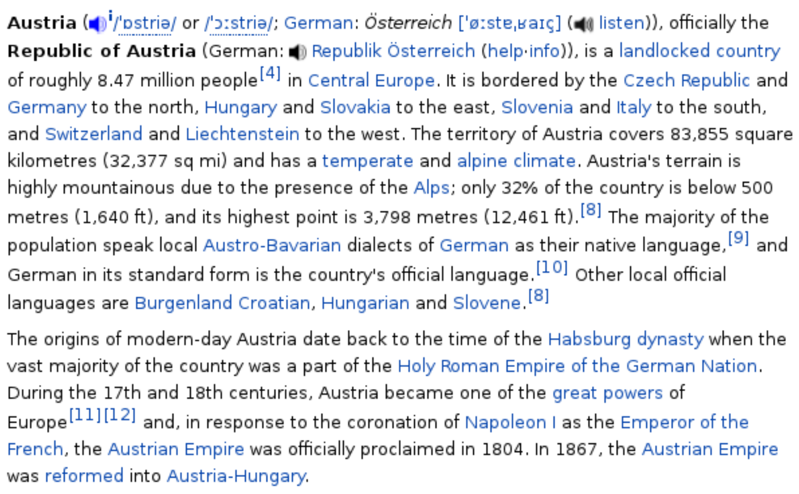
As the web advances to a web of information producers, users are currently faced with two options regarding the way in which information is stored online: (i) to store information as fulltext and (ii) to store information in a fully structured manner. When choosing the first option, information does not have to adhere to any structure, the user is free to use (or rather not use) any arbitrary structure she would like to use for storing the desired information. The main advantage of such an approach is that the user is able to store any kind of information and does not have to cope with any limitations in regards to the structure of the information to be entered. In contrast, the second approach requires the user to adhere the information to a predefined structure (also referred to as "schema"). Such structured storage is especially beneficial in systems or platforms where all information is structured equally. Traditional applications which are based on such a storage paradigm are e.g. (relational) databases in the banking sector or in business, where e.g. catalogs of goods are stored. The products listed in such a catalog all are structured the same way, they e.g. all feature a product number, name, description, price, etc. The main advantage of this approach is that due to the predefined structure, all data operations can be optimized based on the structure and hence, huge amounts of data can be dealt with in an efficient manner. However, trying to store data which does not adhere to the predefined schema might not be possible at all and poses a problem. One possible solution to this problem is extending the predefined structure to meet the new requirements (in relational databases, this would be realized by adding one or more columns to the respective table). However, when having to pursue such a strategy multiple times, the size of the resulting schema increases which potentially leads to a sparse data set as not all information required by the schema might be available for all items. Also, the task of having to adjust the existent structure can be tedious and time-consuming. Thus, structured storage can be highly optimized however it comes at the price of lowered flexibility in regards to schema changes and new requirements to the structure. Such a lack of flexibility also implies that the user may not be able to store all information she wants to store.

In today's online social media platforms one of the primary goals is to encourage users to participate. Such participation can be manifold: communicating with friends, engaging in discussions and also being part of a community which creates and maintains information, like on Wikipedia. Constraining a participating user in regards to the structure of information she wants to store might lead to a lowered motivation and less data entered. The Wikipedia platform[1] mainly features fulltext, as can be seen in Figure 4.1 which depicts an excerpt of the Wikipedia article about Austria[2]. The fulltext of articles enables users to easily add information without having to cope with schema constraints. The community of Wikipedia users is very committed and hence, fulltext information is maintained and modifications are examined closely. However, focused search on such data is hardly possible. Querying Wikipedia for all countries which have more than 5,000,000 inhabitants and an area of less than 80,000 km^2 is hardly possible. Computing the answer to such a query based on fulltext information would require the automatic extraction of the queried facts, which can be a an erroneous and tedious task. Based on these previously extracted facts, all countries fulfilling the query's constraints can be selected.

Due to the shortcomings of both storage approaches described above (fulltext and fully structured), the *semistructured storage paradigm* aims at combining the benefits of both of these approaches. During the last years, the most prominent representative of semistructured data was to model data as triples which feature a subject, predicate and an according object. By using such a

[1]`http://www.wikipedia.org`

[2]`http://en.wikipedia.org/wiki/Austria`



Austria (i/ˈɒstriə/ or /ˈɔːstriə/; German: *Österreich* [ˈøːsteˌʁaɪç] (listen)), officially the **Republic of Austria** (German: Republik Österreich (help·info)), is a landlocked country of roughly 8.47 million people[4] in Central Europe. It is bordered by the Czech Republic and Germany to the north, Hungary and Slovakia to the east, Slovenia and Italy to the south, and Switzerland and Liechtenstein to the west. The territory of Austria covers 83,855 square kilometres (32,377 sq mi) and has a temperate and alpine climate. Austria's terrain is highly mountainous due to the presence of the Alps; only 32% of the country is below 500 metres (1,640 ft), and its highest point is 3,798 metres (12,461 ft).[8] The majority of the population speak local Austro-Bavarian dialects of German as their native language,[9] and German in its standard form is the country's official language.[10] Other local official languages are Burgenland Croatian, Hungarian and Slovene.[8]

The origins of modern-day Austria date back to the time of the Habsburg dynasty when the vast majority of the country was a part of the Holy Roman Empire of the German Nation. During the 17th and 18th centuries, Austria became one of the great powers of Europe[11][12] and, in response to the coronation of Napoleon I as the Emperor of the French, the Austrian Empire was officially proclaimed in 1804. In 1867, the Austrian Empire was reformed into Austria-Hungary.

Figure 4.1.: Fulltext of an Article (Austria) on Wikipedia

triple notation, any real-world object can be described. Austria's number of inhabitants can be modelled as a triple as can be seen in the following:

```
<Austria> <inhabitants> <8,414,638>
```

The information modelled as a triple in the above example features structure which e.g. enables querying for all countries having more than 5,000,000 inhabitants by selecting all triples featuring a predicate "inhabitants" and an according object value higher than 5,000,000. Subject, predicate and object may be chosen freely by the user at the time of storage. However, the downside of this freedom is that different users may choose different predicates for describing semantically equivalent information, e.g., other users could store information about population numbers using the predicate "population". Such a behaviour leads to incomplete search results because when querying the set of triples for all entries featuring the predicate "inhabitants", the search result does not incorporate triples featuring the predicate "population", which semantically also contain information relevant to the stated query. As can be seen, proliferation and heterogeneity of the structure of data are a severe threat for the performance of search facilities based on this data.

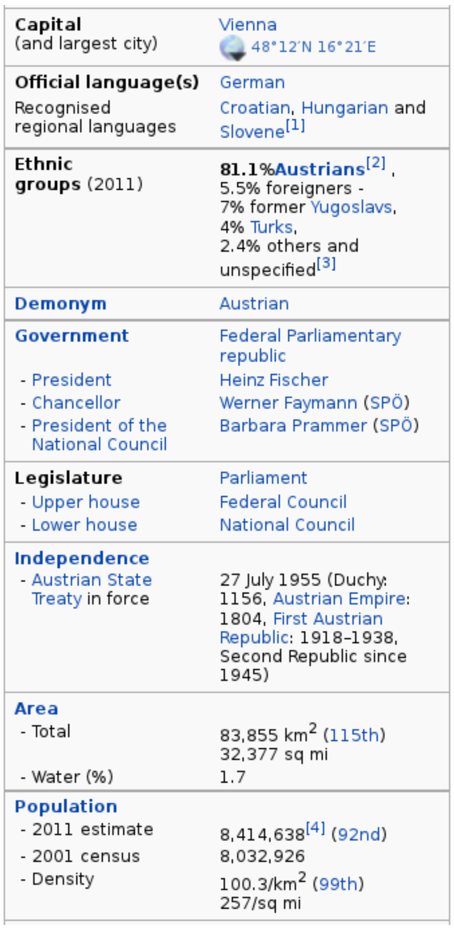
To tackle the problem of a heterogeneous set of predicates and objects, we propose to use a recommender system which guides the user during the process of entering information by providing suitable recommendations for predicates and objects. This approach allows users to still enter any information they would like to enter, however the recommendations provide means to easily enter data adhering to a common structure aiming at a homogeneous data set which provides the basis for efficient search facilities.

In this chapter, we introduce the Snoopy concept, a paradigm for semistructured information systems which facilitates recommender systems in order to create and maintain a homogeneous structure within the stored semistructured information and thus, enabling efficient search facilities. The remainder of this chapter is structured as follows. Section 1 contains a description of semistructured concepts and their characteristics. Section 2 subsequently introduces the Snoopy concept which forms the basis for the presented semistructured information system. Section 3 presents the recommender system and the according algorithms required for an implementation of the Snoopy concept. Subsequently, Section 4 contains the evaluation of the proposed approach. Section 5 describes work closely related to the proposed approach and Section 6 contains a description of future work and concludes this chapter.

4.1. Semistructured Data

Semistructured data combines both the flexibility of not being forced to adhere to a rigid schema and the benefits of some structure present in regards to querying facilities. Semistructured data was firstly discussed in the 90s [16], where Peter Buneman explained the need for such semistructured data by three factors: (i) the need for a flexible data structure for web data in order to be able to query (mostly unstructured) web data, (ii) the need for a flexible data exchange format between databases and (iii) the need for semistructured data as a means for browsing through structured data. To some extent, these requirements are still valid nowadays as there still is a need for a data format which allows for a flexible structure of data enabling powerful query facilities while at the same time not constraining the data to be stored in regards to its structure, as e.g., in an information system like the system proposed in this thesis. Also, in regards to data on the world wide web, facilities for precise queries are somewhat limited. Web search still mostly relies on fulltext search due to the lack of structure within the stored data. Throughout the last years, different projects tried to tackle this problem by providing structured data sets to the public. The Wikipedia platform introduced so-called infoboxes which are tabular aggregations of the most important facts about the subject of the according page. Figure 4.2 depicts a part of the infobox of the Wikipedia page about Austria. As can be seen, such an infobox provides structure as information chunks are provided as key-value pairs, e.g., the fact that Austria's total area is 83,855 km^2 or that it's capital is Vienna. Based on such structured information, complex queries can be evaluated and answered as structure allows for a precise formulation of queries and likewise, the computation of results.

The Resource Description Framework (short: RDF) [64] currently is the most dominant and popular representative of semistructured data. RDF aims at storing information as triples where a triple features a subject, predicate and



Capital (and largest city)	Vienna 48°12′N 16°21′E
Official language(s)	German
Recognised regional languages	Croatian, Hungarian and Slovene[1]
Ethnic groups (2011)	**81.1%Austrians**[2] , 5.5% foreigners - 7% former Yugoslavs, 4% Turks, 2.4% others and unspecified[3]
Demonym	Austrian
Government	Federal Parliamentary republic
- President	Heinz Fischer
- Chancellor	Werner Faymann (SPÖ)
- President of the National Council	Barbara Prammer (SPÖ)
Legislature	Parliament
- Upper house	Federal Council
- Lower house	National Council
Independence	
- Austrian State Treaty in force	27 July 1955 (Duchy: 1156, Austrian Empire: 1804, First Austrian Republic: 1918-1938, Second Republic since 1945)
Area	
- Total	83,855 km^2 (115th) 32,377 sq mi
- Water (%)	1.7
Population	
- 2011 estimate	8,414,638[4] (92nd)
- 2001 census	8,032,926
- Density	100.3/km^2 (99th) 257/sq mi

Figure 4.2.: Infobox on the Wikipedia Article about Austria

an object. Each of these three components are described by a URI (Uniform Resource Identifier[3]). URIs provides means to uniquely identifying arbitrary resources and objects across the whole web. The above mentioned example extracted from a Wikipedia infobox (area and capital of Austria) can be modelled as RDF as can be seen in Listing 4.1. In this case, Austria is the subject of the triple whereas "areakm" and "capital" are used as the predicates of the two respective triples and for the area of Austria, the value is directly specified. In contrast, the capital of Austria is specified as a URI as Vienna is a resource itself and hence, due to the specification of Vienna as a resource, a link is created which can be followed in order to retrieve more information about Vienna. Such a usage of links to other resources leads to a graph of information.

[3]`http://www.w3.org/TR/uri-clarification/`

```
<http://dbpedia.org/resource/Austria> <http://dbpedia.
    org/property/areakm> <83,855>
<http://dbpedia.org/resource/Austria> <http://dbpedia.
    org/property/capital> <http://dbpedia.org/resource/
    Vienna>
```

Listing 4.1: RDF Representation of Facts about Austria

4.2. Snoopy Concept

In the following we present the Snoopy concept, a concept for semistructured information systems which aims at creating and maintaining a homogeneous set of properties and objects within the system. We propose to store information about any arbitrary subject as property-value pairs. Consider the subject of Austria as shown in Listing 4.2, where information about Austria is stored as property-value pairs.

```
Austria
capital: Vienna
inhabitants: 8,414,638
area: 83,855 km2
```

Listing 4.2: Austria, Exemplary Subject

By letting the users store information as property-value pairs belonging to a certain subject, information is implicitly stored as triples by the user, i.e. the stored information can directly be converted to triples as can be seen in Listing 4.3.

```
<Austria> <capital> <Vienna>
<Austria> <inhabitants> <8,414,638>
<Austria> <area> <83,855 km2>
```

Listing 4.3: Information about Austria in Triple Notation

It is important to note that the user is still able to choose arbitrary properties and according values to populate the information stored e.g., about Austria. Such a system allows the user to freely enter information in the form of property-value pairs associated with a given subject without having to adhere to a given schema. Nevertheless, the stored information features a certain amount of structure due to the triple format. However, when considering that many different users stemming from various backgrounds, languages, professions, etc. enter information into such a system, the set of entered properties and objects is bound to get heterogeneous[4]. Such a proliferation of structures is highly disadvantageous in terms of search capabilities as the system features many synonymous properties and objects. The less synonymous and ambiguous information is featured in the data set, the higher the precision and recall of search facilities. Hence, the goal of the Snoopy concept is to avoid a steady proliferation of schemata by providing its users with suitable recommendations regarding both the structure and the content of the information system. Furthermore, we propose to present the user with recommendations for information which she might also want to add to the subject she entering information about. In particular, the Snoopy concept aims at providing the following support mechanisms to users:

- Content-aware structure recommendations
- Content recommendations and semantic refinements
- Auto-completion features

These mechanisms and the underlying algorithms are thoroughly discussed in more detail in the following.

4.2.1. Content-Aware Structure Recommendations

The notion of structure recommendations refers to recommendations for additional properties for the subject the user currently is adding information to. Such recommendations are aimed at providing the user with suggestions for further snippets of information which the user might also be interested in adding to the current subject (along with a respective value), i.e. to point the user to properties which were already used by other users on similarly structured subjects. Hence, the user is encouraged to enter more information as the system "snoops" more information by suggesting further bits of information which the user might want to add. The main benefit of such recommendations is that in the case of a user who accepts (at least some of) these recommendations, the set of properties is not unnecessarily enlarged as already existent properties are recommended and hence, these are re-used, not contributing to a proliferation of properties used within the system.

Consider a user who already entered information about the area, the capital and the population of Austria, resulting in three triples as shown in Listing 4.3. Based on these properties entered on the current subject (Austria), the goal is to find further properties on similar subjects which are suitable for the subject

[4]Furnas *et al.* showed in [31] that if two humans are presented with the same object, the chance of both humans choosing the same name for it is 20%.

the user is currently working on. If e.g., a sufficiently large number of other subjects which also feature the three properties “inhabitants”, “capital” and “area” also contain the property “president” (and the according value), this property is recommended to the user as it is very likely that this property might also be suitable for the current subject. Hence, the user may easily add this property and specify the according value. In particular, the system is searched for the most similar subjects where the similarity of subjects is solely defined by the intersection of the sets of properties on the two subjects. Based on the set of the most similar subjects, the properties used on these subjects are extracted and ranked in order to provide the user with the most suitable and most popular properties. It is important to note that the set of recommended properties is computed on-the-fly and hence gets refined and updated whenever a new property is added to the current subject in order to provide the user with the best suitable recommendations.

4.2.2. Content Recommendations and Semantic Refinements

The second set of recommendations proposed for the Snoopy concept are aimed at creating and maintaining a homogeneous set of values within the information system. This is realized by providing the user with exemplary values for the specified property. If the user e.g. entered the property “country” on a certain subject, the system provided the information that other users entered values like “Austria” or “Italy”. Such recommendations are also used to point users to units and metrics which are commonly used by other users, like for specifying the area of a certain country, the users are provided with the information that other users entered the according information as “100,000 km^2”. This way, the user may be prevented from entering semantically equivalent information in a syntactically different way, e.g. as 100,000,000,000 m^2.

Additionally, the recommendation of values can be exploited for a semantic refinement of the information entered by the user. This is realized by suggesting linking the entered values to already existing subjects in order to be able to uniquely identify objects. This contributes to dissolving ambiguous entries, as e.g., in the case of homonyms. If a user e.g. stated “Cambridge” as the object associated with the property “city”, it is not clear whether the user refers to Cambridge in the United Kingdom or Cambridge in Massachusetts in the U.S.. However, if these two cities have already been entered in the information system, this ambiguity can be resolved by linking the according object to the correct subject. By doing so, the objects gets uniquely identified and moreover, this link can be traversed in order to find more information about this specific object.

4.2.3. Auto-Completion Features

In addition to two types of recommendations as described in the previous sections, the user of the Snoopy concept is also provided with an intelligent auto-completion feature which is aimed at minimizing the amount of entered synonymous properties and values. Consider a user who is about to enter the property "inhabitants". After having started typing, the user is provided with the set of the most popular properties within the system which feature the already entered characters as a substring, e.g. "numberOfInhabitants". This way, the user can be provided with all properties which might be equivalent to the property the user intended to enter. Typically, such recommendations are accepted by users and hence, the set of properties used throughout the system remains stable and homogeneous, as the addition of further synonymous properties is avoided.

4.3. Recommendation Mechanisms

In the following section, the recommendation mechanisms for structure recommendations within the Snoopy concept is explained in detail. The implementation of content recommendations is not discussed in detail as the main focus of this work lies on property recommendations.

The computation of content-aware structure recommendations is based on finding similar subjects which feature properties which are not contained in the current subject yet, i.e. to detect properties which co-occur frequently on similar subjects and are new to the user in the context of the given subject. In particular, we propose to perform a co-occurrence analysis based on the findings in the field of association rule learning [4]. Association rules are typically used for the analysis of transaction data of products, i.e. to detect all items which are frequently bought together in order to optimize the according marketing strategy. A transaction can be considered as a shopping basket of a customer. More formally, a transaction consists of a set of items $I_1, I_2, ..., I_n \in \mathcal{I}$. An association rule can then be defined as $X \Rightarrow Y$, where X is a subset of items $\mathcal{I}$ (the antecedent of the rule) and Y is an item which is not contained in X (the consequent of the rule). In the field of basket analysis, such a rule implies that if a customer bought all items in X, he might also buy Y (with a certain probability estimated by the computed confidence value of the rule).

As for the Snoopy concept, we propose to utilize the set of properties used on a specific subject as the set of items featured in one transaction. Based on such a transaction, the association rules are extracted. Consider an exemplary subject featuring the properties "Inhabitants", "Capital" and "Area". The extracted association rules can be seen in Table 4.1. In order to be able to compute recommendations efficiently for a huge number of subjects and according properties, we propose to only make use of one-element antecedents as this allows for an efficient implementation while at the same time still allows for a fine-grained modeling of co-occurring properties. Furthermore, co-occurrence of items on a single subject can be seen as a symmetric relation of properties (if properties p_a and p_b are featured on the same subject, both the rules $p_a \Rightarrow p_b$ and $p_b \Rightarrow p_a$ hold). Hence, we propose to only store one of the two semantically equivalent rules and thus, view the proposed association rules as pairs of properties co-occurring on a subject. We extend this notion of pairs of co-occurring properties to a triple consisting of the two co-occurring properties and an integer value c such that an association pair can be denoted as (p_a, p_b, c) which describes that the properties p_a and p_b both occur on c subjects within the data set.

Properties	**Association Rules**	**Association Triples**
Inhabitants	Inhabitants ⇒ Capital	(Inhabitants, Capital, 1)
Capital	Inhabitants ⇒ Area	(Capital, Area, 1)
Area	Capital ⇒ Area	(Inhabitants, Area, 1)
	Capital ⇒ Inhabitants	
	Area ⇒ Inhabitants	
	Area ⇒ Capital	
	Area, Inhabitants ⇒ Capital	
	Area, Capital ⇒ Inhabitants	
	Inhabitants, Capital ⇒ Area	

Table 4.1.: Association Rules Example

Algorithm

In the following, our algorithm for the computation of content-aware structure (property) recommendation candidates is presented and explained. The actual algorithm can be seen in Algorithm 3. We make use of (i) the set of association triples stored for all previously entered information in the system denoted as $\mathcal{R}$ and (ii) the set of properties currently used on the current subject S_i, denoted as $\mathcal{P}_{S_i}$ as input for the algorithm. For each of the properties already associated with the current subject, the set of association triples is searched for triples which (i) feature this specific property and (ii) contain a second property which is new in the context of the given subject. If such a triple is detected, the association triple is aligned and added to the set of recommendation candidates. Such an alignment is required as it enables us to identify the antecedent and the consequent of the rules, i.e. we are able to detect which of the two properties is already featured on the current subject and which of the properties is a candidate for later recommendation. The aligned triple is then added to the set $\mathcal{T}$ of recommendation candidates.

```
Input: $\mathcal{P}_{S_i}$, the set of properties associated with the current subject $S_i$
Input: $\mathcal{R}$, the set of association triples of the system
Result: set $\mathcal{T}$ of all structure recommendation candidates for $S_i$
// Initialization
// Set of all association rules leading to new information
$\mathcal{T} \leftarrow \{\ \}$
// Detect all rules featuring a consequent
// Not featured on the current subject
foreach $p_i \in \mathcal{P}_{S_i}$ do
    foreach $(p_a, p_b, c) \in \mathcal{R}$ do
        if $(p_a == p_i \land p_b \notin \mathcal{P}_{S_i})$ then
            $\mathcal{T} \leftarrow \mathcal{T} \cup \{(p_a, p_b, c)\}$
        end
        else if $(p_b == p_i \land p_a \notin \mathcal{P}_{S_i})$ then
            $\mathcal{T} \leftarrow \mathcal{T} \cup \{(p_b, p_a, c)\}$
        end
    end
end
// Return set of recommendation candidates
return $\mathcal{T}$
```

Algorithm 3: Recommendation Candidate Computation

Ranking

The ranking of the computed content-aware structure recommendation candidates is a crucial task as in an information system aiming at allowing users to easily and efficiently enter and manipulate data, a user cannot be presented with a long list of recommended items. Therefore, suitable ranking mechanisms are crucial in order to be able to present the user with the top-k most suitable structure recommendations. We propose two different ranking mechanisms for the computed structure recommendation candidates, which we present in the following.

Generally, the ranking of content-aware structure recommendation is based on the set $\mathcal{T}$ which is computed as shown in Algorithm 3. The set $\mathcal{T}$ consists of association triples (p_a, p_b, c) where the co-occurring properties are all aligned such that p_a is a property which is already used on the specific subject and p_b is not already used and hence a candidate for recommendation.

The first and straight-forward ranking mechanism is based on the count-value c for given association triples. We refer to this ranking method as *Global Popularity Rank*. The algorithm underlying this ranking mechanism is sketched in Algorithm 4. The algorithm clearly depicts that based on the set of recommendation candidates $\mathcal{T}$, the count values c for each association triple (p_a, p_b, c) aggregated for each p_b, i.e. the sum of all count values c of rules having p_b as consequent is computed. Subsequently, we rank the recommended properties in a descending order based on the aggregated count values. This ranking mechanism allows us to rank those properties higher which frequently co-occur with properties already used on the current subject.

```
Input: $\mathcal{T}$, the set of recommendation candidates
Output: L, ranked list of structure recommendations for subject $S_i$
$L := [\,]$
$\mathcal{C} := \{\,\}$
// For all of the rules in $\mathcal{T}$ compute rule counts
foreach $(p_a, p_b, c_i) \in \mathcal{T}$ do
    if $(\exists (p_x, c) \in \mathcal{C},\ where\ p_x == p_b)$ then
        $\mathcal{C} = \mathcal{C} \backslash \{(p_x, c)\}$
        $\mathcal{C} = \mathcal{C} \cup \{(p_x, c + c_i)\}$
    end
    else
        $\mathcal{C} = \mathcal{C} \cup \{(p_b, c_i)\}$
    end
end
// Sort candidates in $\mathcal{C}$ by $c$ descending
L = sort ($\mathcal{C}$, $c$)
return $L$
```

Algorithm 4: Global Popularity Rank

The second proposed ranking mechanism, the so-called *Context-sensitive Rank*, is not directly related to the popularity of co-occurrences between properties. It is rather based on the number of different association triples which imply the recommendation of a certain property p_b, i.e. the number of rules within $\mathcal{T}$ featuring property p_b as consequent. The algorithm underlying this ranking method can be seen in Listing 5. Hence, the more different rules feature a certain property as consequent, the higher the rank of the according property. Thus, this ranking method aims at strengthening the influence of the context of a given subject for the ranking. This ranking is more sensitive to the context of the recommendation as it directly accounts for the properties used on the current subject. In contrast, the global popularity ranking method relies on global popularity counts of association triples which might not resemble the current context.

```
Input: $\mathcal{T}$, the set of recommendation candidates
Output: L, ranked list of structure recommendations for subject $S_i$
L := [ ]
$\mathcal{C}$ := { }
// For all of these rules in $\mathcal{T}$ compute rule counts
foreach $(p_a, p_b, c_i) \in \mathcal{T}$ do
    if $(\exists (p_x, c) \in \mathcal{C},\ where\ p_x == p_b)$ then
        $\mathcal{C} = \mathcal{C} \setminus \{(p_x, c)\}$
        $\mathcal{C} = \mathcal{C} \cup \{(p_x, c+1)\}$
    end
    else
        $\mathcal{C} = \mathcal{C} \cup \{(p_b, 1)\}$
    end
end
// Sort candidates in $\mathcal{C}$ by $c$ descending
L = sort ($\mathcal{C}$, $c$)
return $L$
```

Algorithm 5: Context-sensitive Rank

4.4. Evaluation

The following section describes the evaluation of the proposed recommendation algorithm.

The structure recommendation approach can be seen as a recommendation of good items [39]. As already carved out in Section 3.5, the recommendation of good items is concerned with providing the user with recommendations the user is most likely to accept. As for the Snoopy concept, the user is presented with recommendations for further properties for the given subject. However, the user cannot be presented with a huge number of properties to choose from due to cognitive limitations and hence, the task is to recommend some items. Due to this restriction, the recommendation task of the Snoopy concept can rather be seen as a classification task which aims at classifying the most suitable items for a specific user [51].

We chose to perform the evaluation of the property recommendations offline based on historic data. This is due to the fact that an automated offline evaluation can assess the accuracy of the recommendations in an efficient manner. In particular, the evaluation is implemented by computing recommendations for parts of subjects taken from the data set and subsequently comparing the set of recommended items (properties) to the set of originally used properties on the specific subject.

In the following we firstly present the metrics used for the evaluation and the data set underlying our evaluation. Subsequently, we present the setup of the evaluation, and the evaluation algorithm. Finally, we present the results of the conducted evaluations.

4.4.1. Metrics

As for the metrics used within this evaluations, we utilize similar metrics as already used for the evaluation of the hashtag recommendation approach (described in Section 3.5). However, we adapted the recall measure such that the denominator is defined by the minimum of the number of recommendations and the number of original remaining properties, as shown in Equation 4.1. In this equation, $\#matching\_properties$ holds the number of properties which were correctly recommended. This is due to the fact that e.g., in the case of computing 10 recommendations, computing the recall value for subjects which only feature e.g. 7 properties may result in misleading figures. Even if 7 recommendations would be correct (and hence, all original properties would be recommended), the recall would still be 0.7 due to the fact that the total number of recommendations (and not the number of original properties) is used for the evaluation.

$$recall = \frac{\#matching\_properties}{min(\#recommendations, \#remaining\_properties)} \tag{4.1}$$

4.4.2. Data Set

As for the data set underlying our evaluation procedures, we chose to make use of a well-established data set, namely the DBPedia data set [8], which is publicly available. This data set provides the content of infoboxes in Wikipedia as triples. A deciding factor for choosing this data set for the evaluations is the fact that the infoboxes from which the information was extracted were created manually by users in a collaborative fashion. Theoretically, the infoboxes featured on Wikipedia articles are not restricted in regards to which properties and values are featured in the infobox. However, these infoboxes are manually maintained by the community and hence, manually aligned to a common schema. Further information about the DBpedia data set can also be found in Section 4.5.1.

In the following, the most important facts about the data set are described. In total, 41,119,872 triples are featured in the data set used for the evaluation. The data set features 3,935,676 distinct subjects, 51,289 distinct properties and 9,764,538 distinct object values. On average, each subject features 10.45 property-value pairs. The property occurring most frequently is `http://dbpedia.org/property/wikiPageUsesTemplate`[5] occurring a total of 4,860,285 times within the data set. `relatedInstance` occurs 1,278,081 times and `name` occurs 1,251,559 times. On average, each property occurs 801.73 times. There are two properties which are used only once: `captionImage` and `liveAlbums`.

Characteristic	**Value**
Subjects distinct	3,935,676
Properties distinct	51,289
Values distinct	9,764,538
Triples total	41,119,872
Max. number of property-value pairs on single subject	4,802
Min. number of property-value pairs on single subject	1
Avg. number of property-value pairs on single subject	10.45
Max. number of occurrences of a single property	4,860,285
Min. number of occurrences of a single property	1
Avg. number of occurrences of a single property	801.73
Max. number of occurrences of a single value	481,736
Min. number of occurrences of a single value	1
Avg. number of occurrences of a single value	4.21

Table 4.2.: Overview Data Set

4.4.3. Evaluation Setup

Before we were able to conduct the actual evaluations, a number of preprocessing steps were taken. Firstly, we downloaded a DBpedia dump containing

[5] In the following, the prefix `http://dbpedia.org/property/` is omitted.

41,111,872 triples representing the content of 3,935,676 infoboxes, respectively Wikipedia articles (subjects). We subsequently split this data set into two parts: a test set and a training set. As for the test set, we chose to only make use of subjects which feature at least six properties in order to be able to evaluate a sufficiently large number of properties on a single subject. A total of 1,500,000 infoboxes feature at least six properties and based on these subjects, we randomly chose 10%, i.e. 150,000 infoboxes to be featured in the test data set and hence, were evaluated. All the remaining 3,785,676 infoboxes were added to the training set. Subsequently, the training set was processed and all association pairs were extracted. This resulted in a raw total of 486,000,000 pairs and after having computed the popularity counts for the respective pairs, resulted in 7,800,000 distinct pairs. This set of association pairs serves as the basis for our evaluations.

4.4.4. Evaluation Algorithm

The evaluation was conducted as a Leave-One-Out test [21] (similar to the evaluation which we conducted for the hashtag recommender system in Section 3.5.3). In the case of property recommendations for semistructured information systems, the following steps are taken:

1. Randomly select a subject from the test set.
2. Randomly remove all but three properties from the subject.
3. Use the remaining three properties as input for the recommender system and compute the ranked property recommendations.
4. We propose to evaluate the approach by two different evaluation techniques:
    a) *Top-k evaluation*: We compare the set of top-k recommended properties to the previously removed properties and compute the evaluation metrics.
    b) *Iterative evaluation*: If one or more properties within the set of recommendations match an original property, randomly add one of these properties to the set of remaining properties. As long as newly computed properties match an original property, we take the (new) set of properties and use these at input for a new computation of recommendations. This evaluation aims at evaluating to which amount our approach is able to reconstruct the original infobox.

These steps can also be seen in Algorithm 6 which formally depicts the evaluation algorithm for one randomly chosen subject. As opposed to the evaluation

```
Data: Set $\mathcal{S}$ of all subjects within the test data set
Data: Boolean $topkEval$ determining whether top-k or iterative
      evaluation is done
// initialization
$S_{curr} := \{ \}$ // subject to be evaluated
$\mathcal{P}_{S_{curr}} := \{ \}$ // properties on the current subject
$\mathcal{P}_{input} := \{ \}$ // input properties for rec.  computation
$\mathcal{P}_{rem} := \{ \}$ // remaining properties not reconstructed yet
$\mathcal{R} := \{ \}$ // set of recommended properties
$\mathcal{C} := \{ \}$ // set of correct recommendations
$p :=$ null // temporary property
// get random subject from $\mathcal{S}$, extract properties
$S_{curr} = \text{getRandomSubject}(\mathcal{S})$
$\mathcal{P}_{S_{curr}} = \text{extractProps}(S_{curr})$
// randomly select three properties, compute recs
$\mathcal{P}_{input} \leftarrow \text{removeProperties}(\mathcal{P}_{S_{curr}}, 3)$
$\mathcal{P}_{rem} \leftarrow \mathcal{P}_{S_{curr}} \setminus \mathcal{P}_{input}$
$\mathcal{R} \leftarrow \text{getRecommendations}(\mathcal{P}_{input})$
if $topkEval$ then
    // top-k evaluation
    $\text{computeMetrics}(\mathcal{R}, \mathcal{P}_{rem})$
else
    // iterative evaluation
    while $((\mathcal{R} \cap \mathcal{P}_{rem}) \neq \emptyset)$ do
        $\mathcal{C} \leftarrow \mathcal{R} \cap \mathcal{P}_{rem}$
        $p \leftarrow \text{getRandomProperty}(\mathcal{C})$
        $\mathcal{P}_{input} \leftarrow \mathcal{P}_{input} \cup \{p\}$
        $\mathcal{P}_{rem} \leftarrow \mathcal{P}_{rem} \setminus p$
        $\text{computeMetrics}(\mathcal{R}, \mathcal{P}_{rem})$
        // refine recommendations
        $\mathcal{R} \leftarrow \text{getRecommendations}(\mathcal{P}_{input})$
    end
end
```

Algorithm 6: Evaluation Algorithm

approach for hashtag recommendations, we did not remove all properties as the remaining properties serve as the input for the recommender system (for the hashtag recommendation task, the text with the original hashtags removed served as input for the recommendation computation task). For this initial set of three properties, the property recommendations are computed and stored in the set $\mathcal{R}$. As for the top-k evaluation, this initial set of recommendations is evaluated in regards to recall and precision. In contrast, the iterative evaluation (cf. lines 21–29 in Algorithm 6) is implemented by an iterative process of getting recommendations, randomly choosing one of the correct recommendations and adding this property to the set of properties featured on the test subject. These properties are then re-used as the input for the next iteration step.

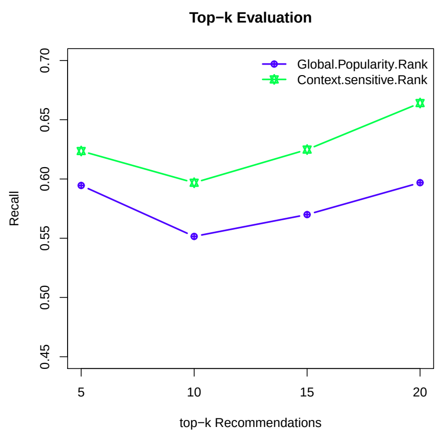
4.4.5. Results

The following section contains the results of the conducted evaluations as previously described.

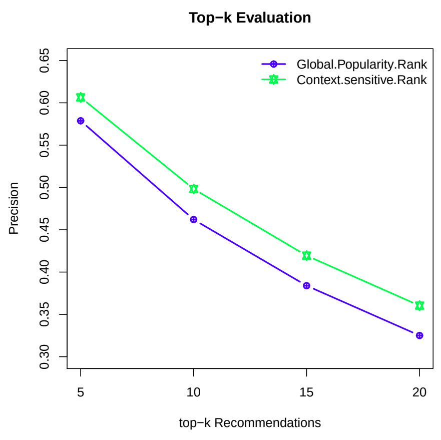
Top-k Recommendation Evaluation

As for the traditional evaluation of precision and recall for top-k recommendations, we chose to perform the evaluation for k being set to 5, 10, 15 and 20. This is due to the fact that showing more than 20 property recommendations does not resemble a realistic setup for recommendations provided in a semistructured information system.

The evaluations showed that the context-sensitive ranking method performs slightly better than the global popularity ranking in terms of recall values. As can be seen in Figure 4.3(a), already for $k = 5$, a recall of more than 62% can be reached when using the context-sensitive ranking, whereas the global popularity ranking method achieves a recall value of 59%. For k being set to 20 recommendations, the recall value can be increased to 66% and 60%, respectively. The minor decrease in performance between k being set to 5 and to 10 properties can be explained by the choice of the evaluation metric and the evaluation method itself. As listed in Table 4.2, each subject features 10.45 properties on average. Hence, when removing all but three properties (as described in the evaluation algorithm, cf. Algorithm 6), the number of original properties left to (correctly) recommend is 7. As pointed out in Section 4.4.1, we adapted the recall measure such that it also incorporates the number of remaining properties on the evaluated subject as well. Consider a subject which features 7 remaining properties. When e.g. computing 5 property recommendations where 3 properties are recommended correctly, the recall value is 0.6. However, when computing 10 property recommendations where 4 properties are recommended correctly, the recall value is 0.57 ($\frac{4}{7}$). Hence, despite increasing the amount of correctly recommended properties, the recall value is lower.



(a) Recall



(b) Precision

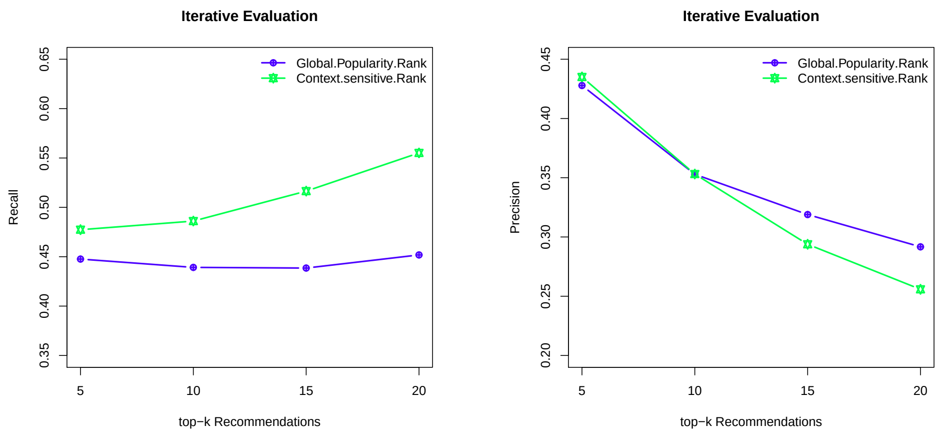
Figure 4.3.: Evaluation for Top-k Recommendations

As for the precision of the structure recommendations, the results are shown in Figure 4.3(b). For a k-value being set to 5, the precision achieved is 60% when ranking the recommendation candidates based on the context-sensitive rank and 57% for a ranking based on the popularity of the according association pairs. Naturally, precision decreases when increasing the number of recommended items presented to the user. Still, at $k = 20$, precision values are still above 30% for both of the proposed ranking methods.

Iterative Recommendation Evaluation

The iterative recommendation evaluation aims at evaluating how an iterative refinement of recommendations affects the performance of the recommendation algorithm. As already carved out in Section 4.4.4, the refinement of recommendations is simulated by iteratively accepting one random property out of the set of correct (originally featured) and suitable properties and subsequently recomputing (refining) the set of property recommendations as long as further recommended properties are suitable.

The evaluations show that the recall performance of the context-sensitive rank method is better than the recall performance of the global popularity ranking method. As can be seen in Figure 4.4(a), a recall value of more than 47% is achieved, increasing to a recall value of 55% when recommending the top-20 items when using the context-sensitive rank. The global popularity ranking method achieves a recall value of 44% when recommending 5 items. However, the recall value hardly increases when increasing the k-value. This can be lead back to the fact that this ranking method is based on global popularity values of co-occurring properties. Hence, when refining the recommendations, the order of recommended properties do not change vastly. In contrast, the context-sensitive rank relies on the number of properties already contained on the subject leading to the recommendation of properties. Hence, it is directly related to the properties associated to the current subject and thus, adding further properties in the course of refinements, entails more comprehensive changes in regards of the ranking of refined recommendations.



(a) Recall

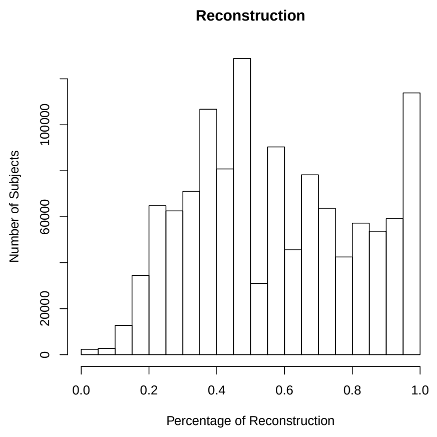
(b) Precision

Figure 4.4.: Evaluation for Iterative Recommendations

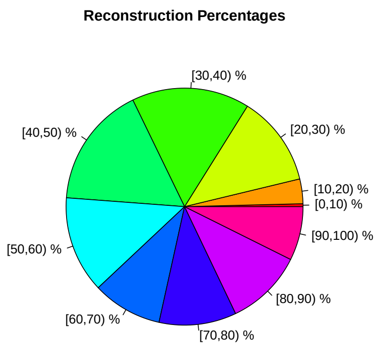
As for the precision of the iteratively computed recommendations, both of the ranking methods perform equally for k being set to 5 and 10. Both methods result in a precision value of 42% for 5 recommended items and 35% for 10 recommended items. However, as k increases to 20%, the global popularity ranking performs better. This sudden change in performance can be lead back to the fact that precision and recall are inversely related measures and as the context-sensitive rank performs better in terms of recall for k being set to 15 and 20, the precision for the same evaluation configuration naturally decreases.

Reconstruction of Subjects

As already described, we also chose to evaluate how many of the subjects within the test data set could be completely reconstructed. As for the ranking method used for this evaluation, we chose to use the context-aware ranking method as it performed better in regards to recall and precision. Furthermore, we made use of the top-20 recommendations. The results of this evaluation can be seen in Figure 4.5. The data set contained a total of 150,000 subjects, each of these featuring at least six properties. Out of these subjects, 16,565 subjects were entirely reconstructed by accepting the recommended properties as stated in the iterative evaluation description above. A total of 90,630 subjects were reconstructed to more than 50%. This sum of subjects amounts to more than 60% of all subjects.



(a) Reconstruction Performance



(b) Percentage of Reconstructed Subjects

Figure 4.5.: Reconstruction Evaluation

4.5. Related Work

The proposed approach is related to various fields of research which incorporate collaborative information systems, Wikis and also the extraction of structured information from various sources on the web.

4.5.1. Structured Data Extraction

The DBpedia project [8] has become a central resource in the Linked Open Data movement. The Linked Open Data movement (LOD) aims at interlinking RDF data sets containing structured information on the web such that a huge interlinked repository of data stemming from various different origins can be queried and used as a whole huge repository. The DBpedia data set consists of structured data which was previously extracted from Wikipedia. Such a structured data set provides facilities to query Wikipedia knowledge in a precise manner as opposed to traditional fulltext search which previously was the only way to search Wikipedia data. Due to the structure, precise and extensive queries can be handled efficiently and hence, all valuable information contained in the Wikipedia encyclopedia can be searched for. Generally, the DBpedia data set is extracted from Wikipedia by taking multiple sources of information into account, as e.g. the Wikipedia category system, internal and external links, infobox templates or geo-coordinate information.

The YAGO (Yet Another Great Ontology) project is also aimed at extracting structured information (entities, concepts and relations) from Wikipedia and providing it to the public. However, in contrast to DBpedia, also WordNet is incorporated for the extraction. Due to this combination Wikipedia and WordNet knowledge, the YAGO ontology achieves both good coverage and accuracy. In particular, all individuals are taken from Wikipedia as it is a very capacious knowledge base. Based on conceptual categories within Wikipedia (those which are not solely used for administrating Wikipedia content), the authors propose to extract information about the type of the according resource, as e.g. the Wikipedia page about Austria is a member of the categories "European countries" and "Member states of the United Nations". From this information, the authors deduce that Austria is part of Europe and a member of the United Nations, respectively. The Wikipedia category system features hierarchy, however this information is mostly used for administrating Wikipedia content and does hardly resemble real-world hierarchies. Therefore, the authors propose to use WordNet and the contained synsets[6] in order to derive subclass-of relationships between concepts featured in YAGO. Synset within WordNet are furthermore used to define semantic equivalence relationships between concepts. Further relations are also deduced from Wikipedia categories, e.g., the fact that Einstein was born in 1879 ("bornInYear": 1879) can be deduced from the category "1879 births". However, all of these extracted facts rely on a rigid rule set and hence, the set of properties which is extracted is rigid and features about 100 relations which were previously defined manually. YAGO2 [44] is an extension to the YAGO knowledge base as it introduces timely and spatial dimensions to the data set. Geolocation information is extracted from Wikipedia and the GeoNames database, timely information is extracted from Wikipedia. The newly enriched data set features 80 million facts about roughly 10 million entities. For both the YAGO and YAGO2 approach the authors state that the quality of the extracted information is nearly human (accuracy is about 95%).

One further approach aiming at automatically extracting structured data is the "Intelligence in Wikipedia" project [112]. It is also based on Wikipedia

[6]WordNet is a lexical database for the English language where words are assigned to one or multiple so-called synsets where one synset features all words having the same meaning. Based on this information, relations between synsets are also provided as e.g. hypernyms or hyponyms for synsets.

and proposes to combine both Information Extraction techniques and mixed initiative techniques aiming at firstly constructing and completing Wikipedia infoboxes and secondly, providing a rich and clean ontology based on the extracted facts. The core of this approach is a self-supervised learning system [114] which uses the existing content of infoboxes as training data. Such a training set if created by taking facts from infoboxes and trying to match such a fact with a sentence within the fulltext of the corresponding Wikipedia article. These labeled sentence form the positive training examples whereas all unlabeled sentences form the negative training examples. For the extraction of facts, the authors make use of two classifiers which are trained by the data set previously extracted. The first classifier is a document classifier which is responsible for identifying articles belonging to the class of articles for which information is to be extracted. The second classifier used for this approach is a sentence classifier. This classifier is used to find sentence which might contain the desired attributes and information in order to be able to construct and complement an infobox. Based on the results of the two classifiers, the extractors are learned by Condition Random Fields. In order to ensure high quality of the automatically extracted facts, the authors propose to make use of a mixed-initiative approach [45]. Hence, the authors propose to ask users of Wikipedia to confirm the correctness of the facts previously extracted. This is done by showing the extracted fact to the user (either by a popup, an icon or highlighted text) asking the user to examine the fact. However, this approach aims at extracting information from already stored texts whereas in contrast our approach aims at supporting the user when inserting new information (as key-value pairs).

A rather recent development in the field of extraction and collaborative curation of structured data is the creation of the Wikidata project[7]. This project is initiated by the Wikimedia foundation (founder of Wikipedia) aimed at creating a common structured knowledge base which is available to the public and is—in contrast to the previously presented knowledge bases—also editable by the public. This knowledge base is aimed at being the central basis for Wikipedia articles, especially for infoboxes, regardless of the language of the article in which the respective infobox is embedded in. Such a central point of information resolves many ambiguities and inconsistencies among various infoboxes stemming from different languages. However, at the time of writing[8], this system does not provide support to its users in terms of guiding the user to homogenize the entered data.

[7]`http://meta.wikimedia.org/wiki/Wikidata/en`

[8]as of October 2012

ExDB (Extraction Database) is a further system aiming at providing structured querying facilities based on (unstructured) web data. This system also extracts facts from web sources in order to create a structure knowledge base which can then be queried in a precise manner. The extraction of facts is based on a probabilistic approach in order to be able to cope with uncertainties arising when automatically extracting facts from the web. Therefore, the system also offers a query language which enables users to formulate probabilistic queries.

4.5.2. Collaborative Information Systems

Freebase [10] is a semistructured information system which currently holds information about more than 23 million different entities[9]. The information contained in the system is harvested from multiple other, publicly available data sets like Wikipedia, MusicBrainz, etc. Additionally, Freebase users may also edit the data contained. In regards to the structure of the data featured in Freebase, each topic (e.g., a page about a certain person) consists of multiple types where each type defines a is-a relationships. A page about Albert Einstein contains the types “Person”, “Award Winner” or “Academic”. Such types may be edited by users, who may also edit and add properties (and according values) belonging to such a type. However, users are not able to edit types which were created by other users. Freebase provides so-called Commons which is a set of topics for which administrators ensured that the topics covered fulfill a certain standard. The types used on the topics within Freebase can be seen as predefined schemata. Due to these schemata, FreeBase does not offer recommendations for further properties as the types are rigidly defined which strongly distinguishes the Freebase approach from the Snoopy concept.

The Cimple/DBLife project [26] is a community information management system which aims at providing a structured information system to a certain community. In order to fill the system with content, information is extracted from the web. The authors propose to create and maintain such a system by firstly making use of a domain expert’s knowledge in order to create a seed set of crawlable websites, desired entities and relationships. Based on this seed set, information is extracted from the web and stored in the information system. As automated extraction traditionally yields errors, the system provides mechanisms enabling users to correct errors, to refine data and to evolve the contained information. However—in contrast to the Snoopy concept—data is aligned after the insertion into the information system.

[9] `http://www.freebase.com/`

Semantic MediaWiki [111] proposes an extension to the traditional Wiki software MediaWiki, which is also used in Wikipedia. This extension of the Wiki markup syntax is used to semantically enhance the content provided in the Wiki. These enhancements include three different types of annotations which can be added to the Wiki markup: (i) the assignment of Wiki articles to certain categories by simply annotating the page with the corresponding category, (ii) the annotation of links between Wiki pages in order to be able to specify the type of the relation and (iii) the annotation of certain facts (attributes) with key-value pairs.

The most distinguishing feature between the presented information systems and the Snoopy concept is that no other system makes use of recommendations already during the time of insertion in order to encourage the user to add already aligned information into the system. To the best of our knowledge, there is no other concept for a semistructured information system which tackles the problem of schema proliferation by supporting the user with recommendation techniques.

4.6. Conclusions and Future Work

In this chapter, we presented a recommender system which enables semistructured, collaborative information systems to create and maintain a homogeneous schema and structure within the stored data. The evaluations conducted showed that the proposed recommendation algorithm is able to reach recall values of 62% when providing a list of five property recommendations. Also, the precision of recommendations reached values of 60% when computing five property recommendations.

In regards to integration of the data produced by an information system based on the Snoopy concept into the Linked Open Data cloud, future work includes an automatic conversion from the internal triples format to standardized RDF. Such a standardization allows for other data sets to interlink with the data provided by the specific information system. This can be realized by matching the entered information to LOD data sources. However, performing this matching after the insertion into the information system can be erroneous and tedious. Hence, a direct integration of LOD sources and the according information into the recommendation computation mechanisms would avoid such problems, especially for uniquely resolving synonymous entities. Still, the integration of such a huge amount of data into the process of computing recommendations is a severe challenge in regards to computability.

Furthermore, the integration of LOD sources can also contribute to a primary boostrapping of the information system. In the course of users entering information in an information system based on the Snoopy concept, recommendations for properties gain higher quality with every new subject that is entered. However, if a new subject describing a new topic or type of entity is entered, there are no similar subjects present within the system as such a new topic may require new properties to be used to describe it. Hence, the recommended properties might not be entirely suitable until a sufficient amount of similar subjects have been entered to the system. Thus, bootstrapping the system with initial data based on LOD sources can contribute to lowering this barrier.

CHAPTER 5

Conclusion

During the last decade, the web has experienced a shift of paradigms as it evolved from a web of information consumers to a web of information prosumers who equally produce and consume online information. The information stemming from such a user behaviour ranges from collaboratively created encyclopedia or collaboratively created and annotated maps to huge repositories of information created by the users of social online media like microblogging platforms. However, this data cannot be fully exploited due to the lack of structure which is a key-enabler for search facilities providing users appropriate instruments to find the desired information as we elaborated on in Section 1.1. Therefore, suitable mechanisms for the creation of a common and homogeneous structure have to be found in order to unlock the full potential of this data.

In this thesis we analysed how recommender systems can be exploited for the introduction and maintenance of structure within such collaboratively created information enabling powerful search facilities for these huge amounts of data. Thus, we formulated the following research questions which we identified in Section 1.1:

RQ1: Can hashtag recommendations be leveraged to enhance structure in microblog entries?

RQ2: Can structure within semistructured data be created and maintained efficiently by content-aware recommendation mechanisms?

In the course of this thesis, the following contributions aiming at answering the two research questions have been made:

Microblogging Environments

As for microblogging environments, we firstly presented a data set crawled from the Twitter microblogging platform and provided a detailed analysis of this data set, especially in regards to the hashtagging behaviour of users and the resulting heterogeneous set of hashtags. Based on these findings, we presented a concept for the recommendation of hashtags in microblogging environments in Section 3.4. The concept is aimed at providing the user with hashtags suitable for the microblog entry the user is currently entering. The proposed recommender system relies on a sufficiently large set of microblog entries which are exploited for the recommendation of hashtags to the user. The basic recommendations are derived from the entries within the data set that are most similar to the entered message. We presented five different similarity measures for microblog entries. The conducted evaluations showed that recommendations based on cosine similarity measures performed best. In order to also be able to incorporate timely factors and user preferences into the recommendations, we also proposed different ranking strategies for the computed hashtag recommendation candidates. The proposed concept was implemented as a prototype which served as a basis for the conducted evaluations. These evaluations showed that an approach which heavily incorporates the similarity of the respective entries for the computation of recommendations leads to suitable and useful recommendations of hashtags and can hence contribute the creation and maintenance of a homogeneous data set.

Semistructured Information Systems

In the field of semistructured information systems, we presented the Snoopy concept which is a semistructured information system which provides guidance in the form recommendations to its users. Users are able to enter information about any arbitrary subject in the form of property-value pairs where the name of the subject, the property and the value can be chosen freely by the user. However, as we identified in Section 4.1, such freedom comes at the risk of creating a heterogeneous set of data which imposes limitations on the search capabilities on this data due to a proliferation of the implicitly contained schemata. We proposed three different types of guidance mechanisms which all are aimed at creating a homogeneous data set providing high search efficiency to its users. We proposed to make use of content-aware structure recommendations, which are the main building block of the Snoopy concept. These recommendations aim at providing the user with suggestions for further properties for which the user might want to add the according value. These recommendations are computed based on an analysis of similarly structured subjects. In Section 4.3 we present the algorithms underlying the content-aware structure recommender system. Furthermore, we presented content-recommendations and semantic refinements, the former providing recommendations for values of already specified properties and the latter aiming at creating links between different triples in order to resolve ambiguous values. Furthermore, we proposed an intelligent auto-completion feature which also aims at prevent the user from entering synonymous properties. The evaluation of these algorithms showed that such recommendations are capable of providing users with useful and adequate recommendations for properties and values. Hence, by accepting these recommendations, the users can contribute to a data set featuring a homogeneous schema which can subsequently be exploited for efficient search facilities.

In conclusion, this dissertation showed that recommendations can be facilitated for the collaborative creation and maintenance of a common and homogeneous structure. We showcased this finding for semistructured information systems and microblogging platforms.

APPENDIX A

Appendix

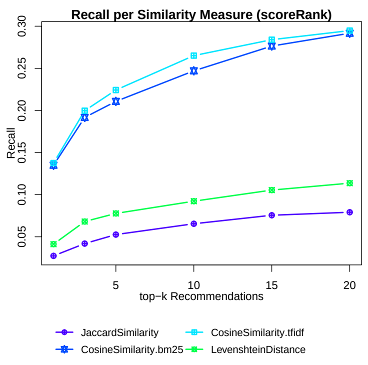
A.1. Twitter API Object

```
{
"_id" : NumberLong("78201820832481280"),
"contributors" : null,
"coordinates" : null,
"created_at" : "Tue␣Jun␣07␣20:49:01␣+0000␣2011",
"entities" :
{
  "hashtags" : [
    {
      "text" : "advertising",
      "indices" : [107, 119]
    },
    {
      "text" : "PR",
      "indices" : [120, 123]
    },
    {
      "text" : "marketing",
        "indices" : [124, 134 ]
      } ],
    "user_mentions" : [ ],
    "urls" : [ ]
},
"favorited" : false,
"geo" : null,
"id_str" : "78201820832481280",
"in_reply_to_screen_name" : null,
"in_reply_to_status_id" : null,
"in_reply_to_status_id_str" : null,
"in_reply_to_user_id" : null,
"in_reply_to_user_id_str" : null,
"place" : null,
"retweet_count" : 0,
"retweeted" : false,
"source" : "web",
"text" : "Working␣on␣an␣assignment:␣What␣do␣you␣see␣as␣the␣biggest␣issue␣facing␣integrated␣marketing␣communications?␣#advertising␣#PR␣#marketing",
"truncated" : false,
"user" : {
    "screen_name" : "nancycdoyle",
    "id_str" : "22835721",
    "is_translator" : false,
    "profile_text_color" : "573dff",
    "followers_count" : 535,
    "url" : "http://pinterest.com/nancycdoyle/",
    "listed_count" : 24,
    "verified" : false,
    "notifications" : null,
    "profile_sidebar_fill_color" : "ffeb87",
    "profile_background_tile" : true,
    "description" : "Just␣a␣girl␣in␣love␣with␣fashion...␣Appreciation␣for␣glossy␣magazines,␣writing,␣design,␣reading,␣orchids,␣travel,␣and␣PR.\r\n\r\n\r\n\r\n\r\n",
    "show_all_inline_media" : false,
    "geo_enabled" : false,
    "favourites_count" : 5,
    "friends_count" : 953,
    "location" : "Halifax,␣Nova␣Scotia,␣Canada",
    "lang" : "en",
    "profile_link_color" : "2a74bf",
  "default_profile_image" : false,
  "profile_sidebar_border_color" : "effa50",
  "contributors_enabled" : false,
  "statuses_count" : 4417,
  "time_zone" : "Santiago",
  "created_at" : "Wed Mar 04 21:16:48 +0000 2009",
  "protected" : false,
  "following" : null,
  "profile_use_background_image" : true,
  "name" : "Nancy C. Doyle",
  "follow_request_sent" : null,
  "profile_background_color" : "c47b79",
  "profile_image_url" : "http://a0.twimg.com/
      profile_images/1278464140/Photo_107_normal.jpg",
  "id" : 22835721,
  "default_profile" : false,
  "profile_background_image_url" : "http://a3.twimg.com/
      profile_background_images/222247664/Picture_5.png",
  "utc_offset" : -14400 }
}
```

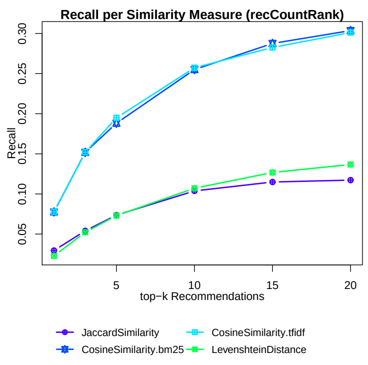
Listing A.1: JSON Object containing information about a Tweet, retrieved via the Twitter API

A.2. Detailed Evaluation Results

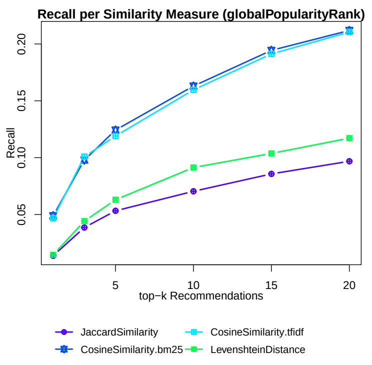
A.2.1. Recall Evaluation



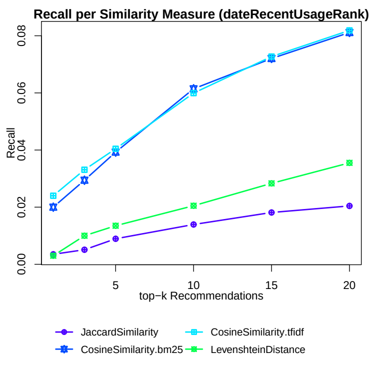
(a) scoreRank



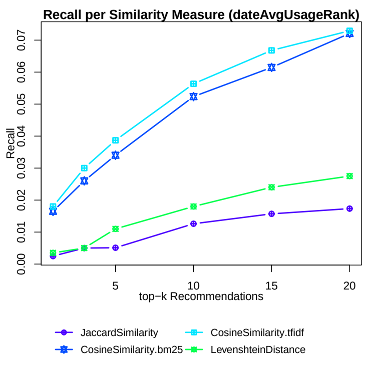
(b) recCountRank



(c) globalPopularityRank



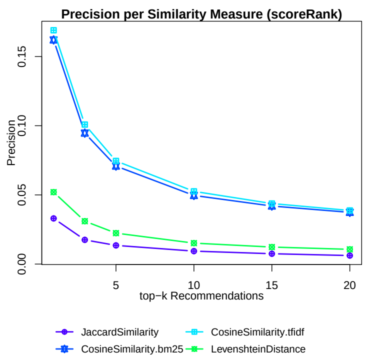
(d) dateRecentUsageRank



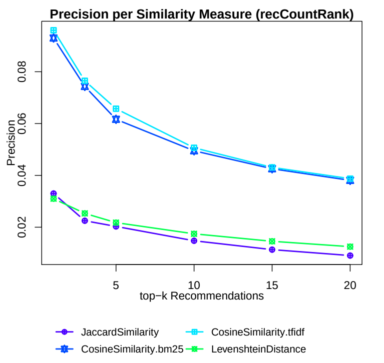
(e) dateAvgUsageRank

Figure A.1.: Recall Values for Top-k Recommendations

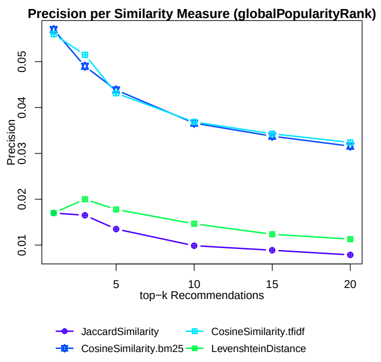
A.2.2. Precision Evaluation



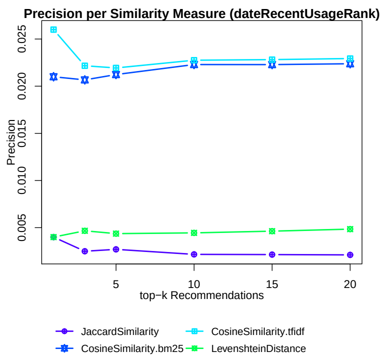
(a) scoreRank



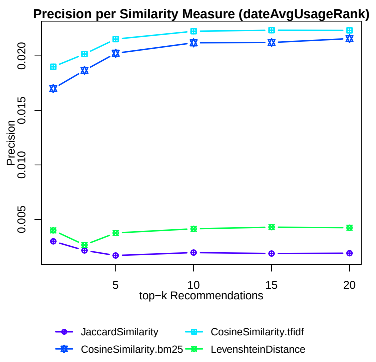
(b) recCountRank



(c) globalPopularityRank



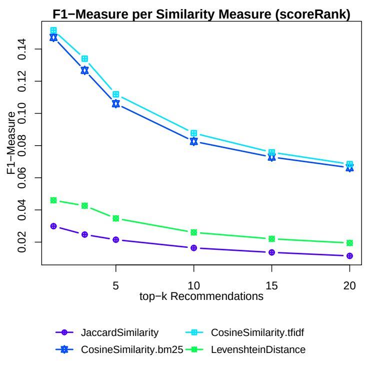
(d) dateRecentUsageRank



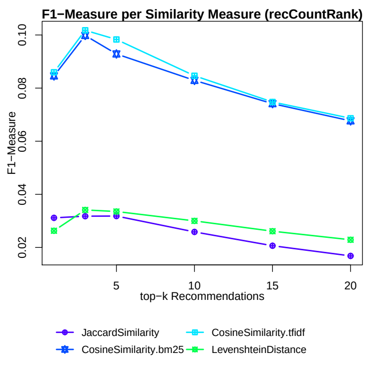
(e) dateAvgUsageRank

Figure A.2.: Precision Values for Top-k Recommendations

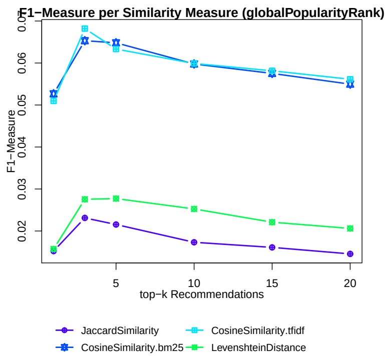
A.2.3. F1 Evaluation



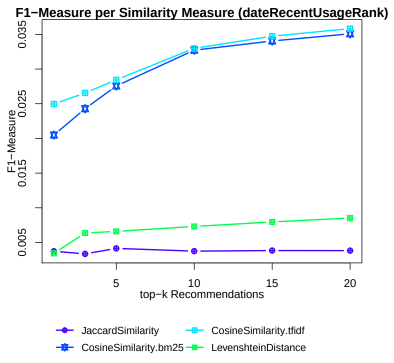
(a) scoreRank



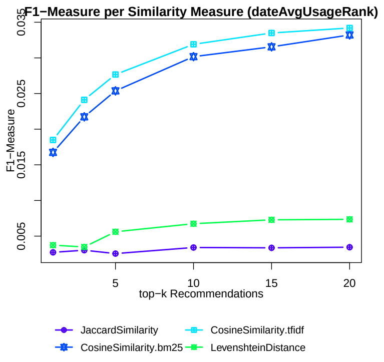
(b) recCountRank



(c) globalPopularityRank



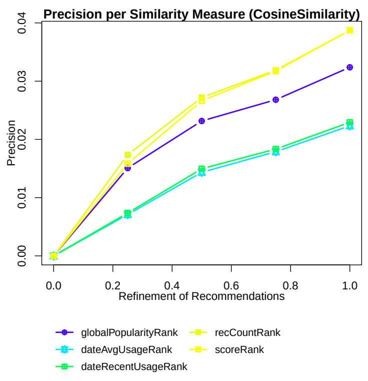
(d) dateRecentUsageRank



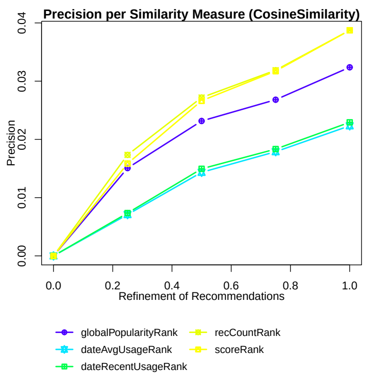
(e) dateAvgUsageRank

Figure A.3.: F1 Values for Top-k Recommendations

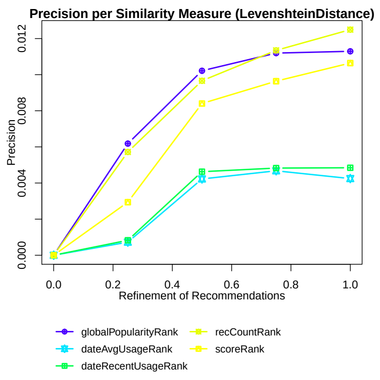
A.2.4. Refinement Precision Evaluation



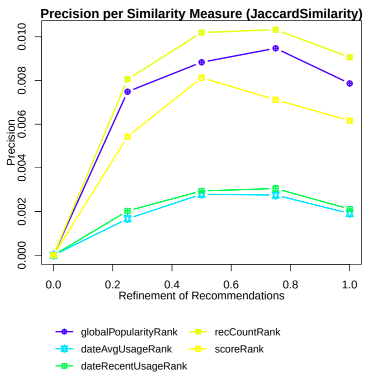
(a) Cosine Similarity (tfidf)



(b) Cosine Similarity (BM25)



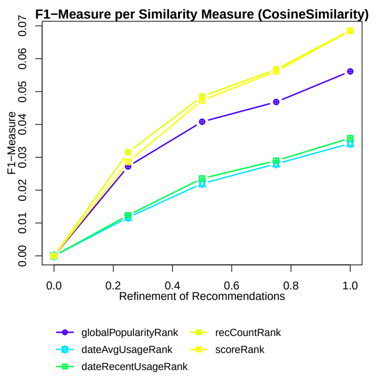
(c) Levenshtein Distance



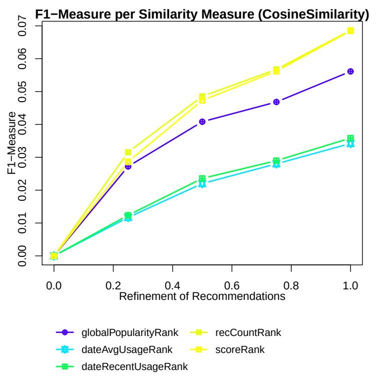
(d) Jaccard Coefficient

Figure A.4.: Refinement Precision Values for Top-k Recommendations

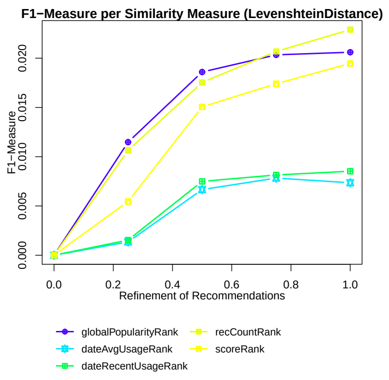
A.2.5. Refinement F1 Evaluation



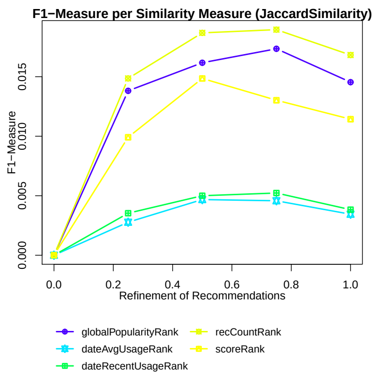
(a) Cosine Similarity (tfidf)



(b) Cosine Similarity (BM25)



(c) Levenshtein Distance



(d) Jaccard Coefficient

Figure A.5.: Refinement F1 Values for Top-k Recommendations

A.2.6. Hybrid Ranking Evaluation

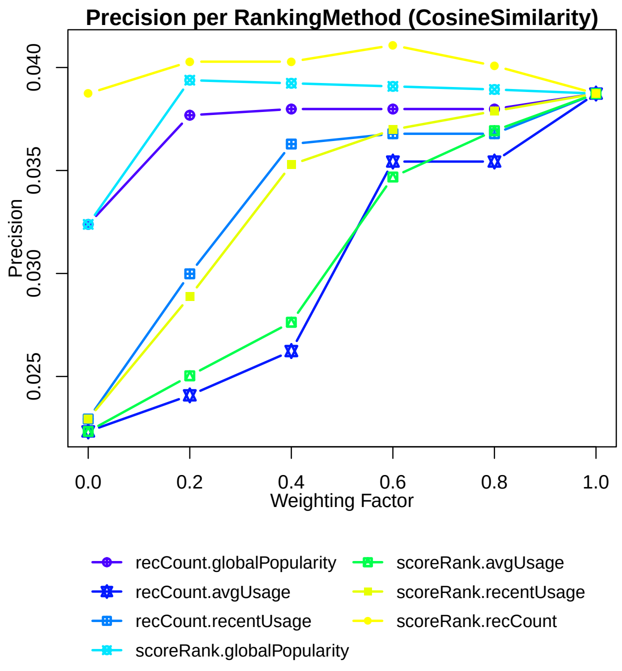


Figure A.6.: Precision for Hybrid Ranking Strategies (Similarity Measure: Cosine Similarity w/ BM25 Weighting, Top-20 Recommendations)

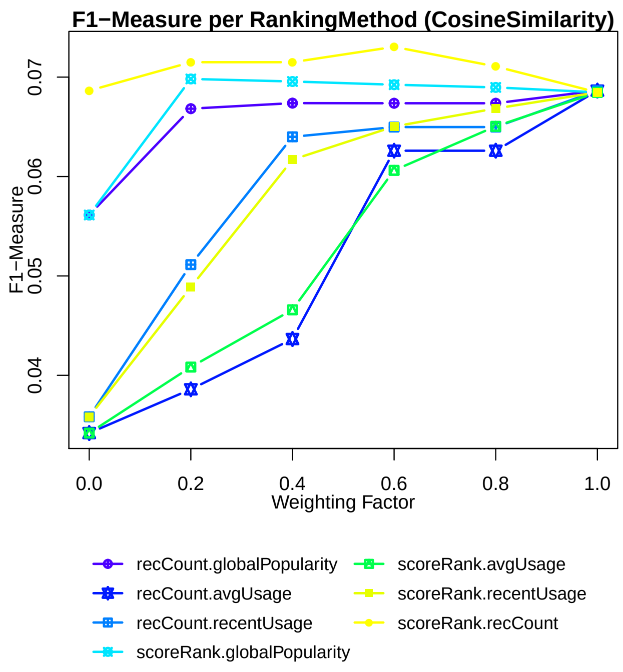


Figure A.7.: F1 for Hybrid Ranking Strategies (Similarity Measure: Cosine Similarity w/ BM25 Weighting, Top-20 Recommendations)

List of Figures

Bibliography

[1] F. Abel, I. Celik, G.-J. Houben, and P. Siehndel. Leveraging the Semantics of Tweets for Adaptive Faceted Search on Twitter. In *The Semantic Web – ISWC 2011*, volume 7031 of *Lecture Notes in Computer Science*, pages 1–17. Springer, Berlin, Heidelberg, New York, 2011.

[2] F. Abel, Q. Gao, G. Houben, and K. Tao. Analyzing User Modeling on Twitter for Personalized News Recommendations. In *Proceedings of the 19th International Conference on User Modeling, Adaption, and Personalization*, UMAP'11, pages 1–12, Berlin, Heidelberg, New York, 2011. Springer.

[3] G. Adomavicius and A. Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[4] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases*, pages 487–499, Burlington, MA, USA, 1994. Morgan Kaufmann.

[5] M. Ames and M. Naaman. Why we Tag: Motivations for Annotation in Mobile and Online Media. In *CHI'07: Proceedings of the SIGCHI*

Conference on Human Factors in Computing Systems, pages 971–980, New York, NY, USA, 2007. ACM.

[6] M. Armentano, D. Godoy, and A. Amandi. Recommending Information Sources to Information Seekers in Twitter. In *International Workshop on Social Web Mining, Co-located with IJCAI 2011*, pages 41–49. Available online at `http://users.cecs.anu.edu.au/~sguo/swm.html`, accessed 2012-10-24.

[7] S. Asur, B. A. Huberman, G. Szabó, and C. Wang. Trends in Social Media: Persistence and Decay. In *Proceedings of the Fifth International Conference on Weblogs and Social Media*, Palo Alto, CA, USA, 2011. Association for the Advancement of Artificial Intelligence (AAAI).

[8] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A Nucleus for a Web of Open Data. In *6th Int'l Semantic Web Conference*, pages 11–15, Berlin, Heidelberg, New York, 2007. Springer.

[9] M. Balabanović and Y. Shoham. Fab: Content-based, Collaborative Recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[10] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250, New York, NY, USA, 2008. ACM.

[11] D. Bollen, B. Knijnenburg, M. Willemsen, and M. Graus. Understanding Choice Overload in Recommender Systems. In *Proceedings of the fourth ACM Conference on Recommender Systems*, pages 63–70, New York, NY, USA, 2010. ACM.

[12] J. Bollen, H. Mao, and X. Zeng. Twitter Mood Predicts the Stock Market. *Journal of Computational Science*, 2(1):1–8, 2011.

[13] D. Boyd, S. Golder, and G. Lotan. Tweet, Tweet, Retweet: Conversational Aspects of Retweeting on Twitter. In *Proceedings of the 43rd Hawaii International International Conference on Systems Science (HICSS-43 2010)*, pages 1–10, Piscataway, NJ, USA, 2010. IEEE Computer Society.

[14] J. Breese, D. Heckerman, and C. Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the four-*

teenth Conference on Uncertainty in Artificial Intelligence, pages 43–52, Burlington, MA, USA, 1998. Morgan Kaufmann Publishers Inc.

[15] A. Bruns and J. E. Burgess. The Use of Twitter Hashtags in the Formation of Ad Hoc Publics. In *Proceedings of the 6th European Consortium for Political Research General Conference*, pages 1–9, University of Iceland, Reykjavik, 2011.

[16] P. Buneman. Semistructured Data. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 117–121, New York, NY, USA, 1997. ACM.

[17] R. Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[18] S. Carter, E. Tsagkias, and W. Weerkamp. Twitter Hashtags: Joint Translation and Clustering. In *3rd International Conference on Web Science (WebSci 2011)*, pages 1–3, New York, NY, USA, 2011. ACM.

[19] J. Chen, R. Nairn, L. Nelson, M. Bernstein, and E. Chi. Short and Tweet: Experiments on Recommending Content from Information Streams. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems*, pages 1185–1194, New York, NY, USA, 2010. ACM.

[20] C. Cleverdon and M. Keen. *Factors determining the Performance of Indexing Systems*. College of Aeronautics, Indiana University, Indiana, 1966.

[21] P. Cremonesi, R. Turrin, E. Lentini, and M. Matteucci. An Evaluation Methodology for Collaborative Recommender Systems. In *Proceedings of the International Conference on Automated Solutions for Cross Media Content and Multi-channel Distribution, AXMEDIS'08*, pages 224–231, Piscataway, NJ, USA, 2008. IEEE Computer Society.

[22] E. Cunha, G. Magno, G. Comarela, V. Almeida, M. A. Gonçalves, and F. Benevenuto. Analyzing the Dynamic Evolution of Hashtags on Twitter: a Language-based Approach. In *Proceedings of the Workshop on Languages in Social Media*, LSM '11, pages 58–65, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[23] F. J. Damerau. A Technique for Computer Detection and Correction of Spelling Errors. *Communications of the ACM*, 7(3):171–176, 1964.

[24] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation - Volume 6*, OSDI'04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[25] H. del Olmo and E. Gaudioso. Evaluation of Recommender Systems: A New Approach. *Expert Systems with Applications*, 35(3):790–804, 2008.

[26] P. DeRose, W. Shen, F. Chen, A. Doan, and R. Ramakrishnan. Building Structured Web Community Portals: A Top-down, Compositional, and Incremental Approach. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 399–410. VLDB Endowment, 2007.

[27] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[28] D. A. Easley and J. M. Kleinberg. *Networks, Crowds, and Markets - Reasoning About a Highly Connected World.* Cambridge University Press, New York, NY, USA, 2010. (p. 482ff).

[29] M. Efron. Hashtag Retrieval in a Microblogging Environment. In *SIGIR '10: Proceedings of the 33rd Annual ACM Conference on Research and Development in Information Retrieval*, pages 787–788, New York, NY, USA, 2010. ACM.

[30] M. Efron. Information Search and Retrieval in Microblogs. *Journal of the American Society for Information Science and Technology (JASIST)*, 62(6):996–1008, 2011.

[31] G. Furnas, T. Landauer, L. Gomez, and S. Dumais. The Vocabulary Problem in Human-System Communication. *Communications of the ACM*, 30(11):964–971, 1987.

[32] P. Ganesan, H. Garcia-Molina, and J. Widom. Exploiting Hierarchical Domain Structure to Compute Similarity. *ACM Transactions on Information Systems (TOIS)*, 21(1):64–93, 2003.

[33] S. Garcia Esparza, M. O'Mahony, and B. Smyth. Towards Tagging and Categorization for Micro-Blogs. In *21st National Conference on Artificial Intelligence and Cognitive Science, AICS*, 2010.

[34] W. Gassler and E. Zangerle. Recommendation-Based Evolvement of Dynamic Schemata in Semistructured Information Systems. In *Proceedings of the 22nd Workshop Grundlagen von Datenbanken (GvDB 2010)*. CEUR-WS.org, ISSN 1613-0073, Vol. 581, available online `http://ceur-ws.org/Vol-581/gvd2010_3_3.pdf`, accessed 2012-10-24, urn:nbn:de:0074-581-7, 2010.

[35] W. Gassler, E. Zangerle, and G. Specht. The Snoopy Concept: Fighting Heterogeneity in Semistructured and Collaborative Information Systems by using Recommendations. In *Proceedings of the 2011 International Conference on Collaboration Technologies and Systems*, pages 61–68, Piscataway, NJ, USA, 2011. IEEE Computer Society.

[36] W. Gassler, E. Zangerle, M. Tschuggnall, and G. Specht. SnoopyDB: Narrowing the Gap between Structured and Unstructured Information using Recommendations. In *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia (HT), Toronto, Ontario, Canada*, pages 271–272, New York, NY, USA, 2010. ACM.

[37] J. Gemmell, T. Schimoler, M. Ramezani, and B. Mobasher. Adapting K-Nearest Neighbor for Tag Recommendation in Folksonomies. In *Proceedings of the 7th Workshop on Intelligent Techniques for Web Personalization & Recommender Systems (ITWP'09) in conjunction with the 21st International Joint Conference on Artificial Intelligence - IJCAI 2009*, pages 51–62. CEUR-WS.org, ISSN 1613-0073, Vol. 528, available online `http://ceur-ws.org/Vol-528/paper8.pdf`, accessed 2012-10-24, 2009.

[38] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[39] A. Gunawardana and G. Shani. A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *The Journal of Machine Learning Research*, 10:2935–2962, 2009.

[40] J. Hannon, M. Bennett, and B. Smyth. Recommending Twitter Users to Follow Using Content and Collaborative Filtering Approaches. In *RecSys '10: Proceedings of the fourth ACM Conference on Recommender systems*, pages 199–206, New York, NY, USA, 2010. ACM.

[41] M. Hepp. HyperTwitter: Collaborative Knowledge Engineering via Twitter Messages. *Knowledge Engineering and Management by the Masses*, pages 451–461, 2010.

[42] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):53, 2004.

[43] W. Hill and L. Terveen. Using Frequency-of-Mention in Public Conversations for Social Filtering. In *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work*, CSCW '96, pages 106–112, New York, NY, USA, 1996. ACM.

[44] J. Hoffart, F. Suchanek, K. Berberich, E. Lewis-Kelham, G. De Melo, and G. Weikum. YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and many Languages. In *Proceedings of the 20th International Conference Companion on World Wide Web*, pages 229–232, New York, NY, USA, 2011. ACM.

[45] R. Hoffmann, S. Amershi, K. Patel, F. Wu, J. Fogarty, and D. Weld. Amplifying Community Content Creation with Mixed Initiative Information Extraction. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems*, pages 1849–1858, New York, NY, USA, 2009. ACM.

[46] C. Honeycutt and S. C. Herring. Beyond Microblogging: Conversation and Collaboration via Twitter. In *Proceedings of Hawaii Conference on System Sciences*, pages 1–10, Piscataway, NJ, USA, 2009. IEEE Computer Society.

[47] L. Hong and B. Davison. Empirical Study of Topic Modeling in Twitter. In *Proceedings of the First Workshop on Social Media Analytics*, pages 80–88, New York, NY, USA, 2010. ACM.

[48] J. Huang, K. Thornton, and E. Efthimiadis. Conversational Tagging in Twitter. In *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia*, pages 173–178, New York, NY, USA, 2010. ACM.

[49] B. Huberman, D. Romero, and F. Wu. Social Networks that Matter: Twitter under the Microscope. *First Monday*, 14(1):8, 2009.

[50] D. Irani, S. Webb, C. Pu, and K. Li. Study of Trend-Stuffing on Twitter through Text Classification. In *Proceedings of the 7th annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference 2010*. Available online at `http://ceas.cc/2010/papers/Paper2013.pdf`, accessed 2012-10-24.

[51] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems: An Introduction.* Cambridge University Press, Cambridge, UK, 2011.

[52] R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag Recommendations in Folksonomies. In *Knowledge Discovery in Databases: Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 506–514, Berlin, Heidelberg, New York, 2007. Springer.

[53] A. Java, X. Song, T. Finin, and B. Tseng. Why we Twitter: Understanding Microblogging Usage and Communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis*, pages 56–65, New York, NY, USA, 2007. ACM.

[54] D. Kim, Y. Jo, I.-C. Moon, and A. Oh. Analysis of Twitter Lists as a Potential Source for Discovering Latent Characteristics of Users. In *Proceedings of the Workshop on Microblogging at the ACM Conference on Human Factors in Computer Systems. (CHI Extended Abstracts 2010)*, New York, NY, USA, 2010. ACM.

[55] S. Kinsella, M. Wang, J. Breslin, and C. Hayes. Improving Categorisation in Social Media using Hyperlinks to Structured Data Sources. *The Semanic Web: Research and Applications*, pages 390–404, 2011.

[56] K. Kireyev, L. Palen, and K. Anderson. Applications of Topics Models to Analysis of Disaster-related Twitter Data. In *Proceedings of the Workshop on Applications for Topic Models: Text and Beyond (co-located with the International Conference on Neural Information Processing Systems).* available online at `http://nips2009.topicmodels.net`, accessed 2012-10-24.

[57] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM*, 40(3):77–87, 1997.

[58] E. Kouloumpis, T. Wilson, and J. Moore. Twitter Sentiment Analysis: The Good the Bad and the OMG! In *Proceedings of the fifth International AAAI Conference on Weblogs and Social Media*, pages 538–541, Palo Alto, CA, USA, 2011. Association for the Advancement of Artificial Intelligence (AAAI).

[59] B. Krishnamurthy, P. Gill, and M. Arlitt. A few Chirps about Twitter. In *Proceedings of the first Workshop on Online Social Networks*, WOSN

’08, pages 19–24, New York, NY, USA, 2008. ACM.

[60] H. Kwak, C. Lee, H. Park, H. Chun, and S. Moon. Novel Aspects coming from the Directionality of Online Relationships: a Case Study of Twitter. *SIGWEB Newsletter*, pages 5:1–5:4, 2011.

[61] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a Social Network or a News Media? In *Proceedings of the 19th International Conference on World Wide Web*, pages 591–600, New York, NY, USA, 2010. ACM.

[62] S. M. Kywe, T.-A. Hoang, E.-P. Lim, and F. Zhu. On Recommending Hashtags in Twitter Networks. In *SocInfo*, volume 7710 of *Lecture Notes in Computer Science*, pages 337–350. Springer, 2012.

[63] D. Laniado and P. Mika. Making Sense of Twitter. In *International Semantic Web Conference*, volume 6496 of *Lecture Notes in Computer Science*, pages 470–485, Berlin, Heidelberg, New York, 2010. Springer.

[64] O. Lassila and R. R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation, W3C, February 1999.

[65] J. Lehmann, B. Gonçalves, J. J. Ramasco, and C. Cattuto. Dynamical Classes of Collective Attention in Twitter. In *Proceedings of the 21st International Conference on World Wide Web*, pages 251–260, New York, NY, USA, 2012. ACM.

[66] J. Letierce, A. Passant, J. Breslin, and S. Decker. Understanding how Twitter is Used to Widely Spread Scientific Messages. In *Proceedings of the WebSci10: Extending the Frontiers of Society On-Line*. available online at `http://journal.webscience.org/view/events/WebSci10=3A_Extending_the_Frontiers_of_Society_On-Line`, accessed 2012-10-24, 2010.

[67] V. Levenshtein. Binary Codes with Correction for Deletions and Insertions of the Symbol 1. *Problemy Peredachi Informatsii*, 1(1):12–25, 1965.

[68] G. Linden, B. Smith, and J. York. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, 7:76–80, 2003.

[69] M. Lipczak and E. Milios. Learning in Efficient Tag Recommendation. In *Proceedings of the fourth ACM Conference on Recommender Systems*, pages 167–174, New York, NY, USA, 2010. ACM.

[70] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008.

[71] M. McPherson, L. Smith-Lovin, and J. Cook. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology*, 27:415–444, 2001.

[72] D. Metzler, S. Dumais, and C. Meek. Similarity Measures for Short Segments of Text. In *Proceedings of the 29th European Conference on IR Research, ECIR 2007*, volume 4425 of *Lecture Notes in Computer Science*, pages 16–27, Berlin, Heidelberg, New York, 2007. Springer.

[73] G. Miller. The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information. *Psychological Review*, 63(2):81, 1956.

[74] B. Mobasher. Recommender Systems. *Künstliche Intelligenz, Special Issue on Web Mining*, 3:41–43, 2007.

[75] E. Mustafaraj and P. Metaxas. From Obscurity to Prominence in Minutes: Political Speech and Real-Time Search. In *Proceedings of the WebSci 2010: Extending the Frontiers of Society On-Line*. available online at `http://journal.webscience.org/view/events/WebSci10=3A_Extending_the_Frontiers_of_Society_On-Line`, accessed 2012-10-24.

[76] M. Naaman, J. Boase, and C. Lai. Is it Really About Me?: Message Content in Social Awareness Streams. In *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work*, pages 189–192, New York, NY, USA, 2010. ACM.

[77] R. Y. Nakamoto, S. Nakajima, J. Miyazaki, S. Uemura, H. Kato, and Y. Inagaki. Reasonable Tag-based Collaborative Filtering for Social Tagging Systems. In *Proceedings of the 2nd ACM Workshop on Information Credibility on the Web*, WICOW '08, pages 11–18, New York, NY, USA, 2008. ACM.

[78] P. Nasirifard and C. Hayes. Tadvise: a Twitter Assistant based on Twitter Lists. In *Social Informatics*, volume 6984 of *Lecture Notes in*

Computer Science, pages 153–160, Berlin, Heidelberg, New York, 2011. Springer.

[79] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab, Stanford, CA, USA, 1999.

[80] A. Pak and P. Paroubek. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC'10)*, pages 1320–1326, Valletta, Malta, 2010. European Language Resources Association (ELRA).

[81] A. Passant, T. Hastrup, U. Bojars, and J. Breslin. Microblogging: A Semantic Web and Distributed Approach. In *4th Workshop on Scripting for the Semantic Web co-located with ESWC 2008*. CEUR-WS.org, ISSN 1613-0073, Vol. 368, available online at `http://ceur-ws.org/Vol-368/`, accessed 2012-10-24, urn:nbn:de:0074-368-11.

[82] M. Pazzani and D. Billsus. Content-Based Recommendation Systems. In *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 325–341, Berlin, Heidelberg, New York, 2007. Springer.

[83] M. J. Pazzani. A Framework for Collaborative, Content-Based and Demographic Filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.

[84] O. Phelan, K. McCarthy, and B. Smyth. Using Twitter to Recommend Real-Time Topical News. In *Proceedings of the third ACM Conference on Recommender Systems*, RecSys '09, pages 385–388, New York, NY, USA, 2009. ACM.

[85] L. Potts, J. Seitzinger, D. Jones, and A. Harrison. Tweeting Disaster: Hashtag Constructions and Collisions. In *Proceedings of the 29th ACM International Conference on Design of Communication*, pages 235–240, New York, NY, USA, 2011. ACM.

[86] D. Ramage, S. Dumais, and D. Liebling. Characterizing Microblogs with Topic Models. In *International AAAI Conference on Weblogs and Social Media*, Palo Alto, CA, USA, 2010. Association for the Advancement of Artificial Intelligence (AAAI).

[87] P. Resnick and H. Varian. Recommender Systems. *Communications of the ACM*, 40(3):56–58, 1997.

[88] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, Berlin, Heidelberg, New York, 2011.

[89] K. Riemer and A. Richter. Tweet Inside: Microblogging in a Corporate Context. In *Proceedings 23rd Bled eConference eTrust: Implications for the Individual, Enterprises and Society*, pages 1–17, 2010.

[90] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proceedings of the Text Retrieval Conference (TREC)*, pages 109–126, Gaithersburg, MD, USA, 1994. National Institute of Standards and Technology.

[91] D. Romero, B. Meeder, and J. Kleinberg. Differences in the Mechanics of Information Diffusion across Topics: Idioms, Political Hashtags, and Complex Contagion on Twitter. In *Proceedings of the 20th International Conference on World wide web*, pages 695–704, New York, NY, USA, 2011. ACM.

[92] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors. In *Proceedings of the 19th International Conference on World Wide Web*, pages 851–860, New York, NY, USA, 2010. ACM.

[93] G. Salton and C. Buckley. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[94] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM.

[95] J. Schafer, J. Konstan, and J. Riedl. Recommender Systems in E-Commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 158–166, New York, NY, USA, 1999. ACM.

[96] M. Schedl. #nowplaying Madonna: a Large-Scale Evaluation on Estimating Similarities between Music Artists and between Movies from Microblogs. *Information Retrieval*, pages 1–35, 2012.

[97] S. Sen, J. Vig, and J. Riedl. Tagommenders: Connecting Users to Items through Tags. In *Proceedings of the 18th International Conference on World Wide Web*, pages 671–680, New York, NY, USA, 2009. ACM.

[98] U. Shardanand and P. Maes. Social Information Filtering: Algorithms for Automating Word of Mouth. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[99] B. Sigurbjörnsson and R. Van Zwol. Flickr Tag Recommendation based on Collective Knowledge. In *Proceeding of the 17th International Conference on World Wide Web*, pages 327–336, New York, NY, USA, 2008. ACM.

[100] G. Specht and T. Kahabka. Information Filtering and Personalization in Databases Using Gaussian Curves. In *Proceedings of the 2000 International Symposium on Database Engineering & Applications*, pages 16–24, Piscataway, NJ, USA, 2000. IEEE Computer Society.

[101] E. Spertus, M. Sahami, and O. Buyukkokten. Evaluating Similarity Measures: a Large-Scale Study in the Orkut Social Network. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 678–684, New York, NY, USA, 2005. ACM.

[102] B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, and M. Demirbas. Short Text Classification in Twitter to Improve Information Filtering. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 841–842, New York, NY, USA, 2010. ACM.

[103] M. Stankovic, M. Rowe, and P. Laublet. Mapping Tweets to Conference Talks: A Goldmine for Semantics. In *Proceedings of the Workshop on Social Data on the Web 2010, Co-located with ISWC 2010*. CEUR-WS.org, ISSN 1613-0073, Vol. 664, available online at `http://ceur-ws.org/Vol-664/`, accessed 2012-10-24, urn:nbn:de:0074-664-4.

[104] M. Steyvers and T. Griffiths. Probabilistic Topic Models. *Handbook of Latent Semantic Analysis*, 427(7):424–440, 2007.

[105] X. Su and T. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, 2009:4, 2009.

[106] J. Teevan, D. Ramage, and M. Morris. # TwitterSearch: a Comparison of Microblog Search and Web Search. In *Proceedings of the fourth ACM International Conference on Web Search and Data Mining*, pages 35–44, New York, NY, USA, 2011. ACM.

[107] K. Thomas, C. Grier, D. Song, and V. Paxson. Suspended Accounts in Retrospect: an Analysis of Twitter Spam. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, pages 243–258, New York, NY, USA, 2011. ACM.

[108] O. Tsur and A. Rappoport. What's in a hashtag?: content based prediction of the spread of ideas in microblogging communities. In *WSDM*, pages 643–652, New York, NY, USA, 2012. ACM.

[109] A. Tumasjan, T. Sprenger, P. Sandner, and I. Welpe. Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, pages 178–185, Palo Alto, CA, USA, 2010. AAAI Press.

[110] S. Vieweg, A. Hughes, K. Starbird, and L. Palen. Microblogging during Two Natural Hazards Events: What Twitter may Contribute to Situational Awareness. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems*, pages 1079–1088, New York, NY, USA, 2010. ACM.

[111] M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer. Semantic wikipedia. In *Proceedings of the 15th International Conference on World Wide Web*, pages 585–594, New York, NY, USA, 2006. ACM.

[112] D. S. Weld, F. Wu, E. Adar, S. Amershi, J. Fogarty, R. Hoffmann, K. Patel, and M. Skinner. Intelligence in Wikipedia. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008*, pages 1609–1614, Palo Alto, CA, USA, 2008. AAAI Press.

[113] J. Weng, E. Lim, J. Jiang, and Q. He. Twitterrank: Finding Topic-Sensitive Influential Twitterers. In *Proceedings of the third ACM International Conference on Web Search and Data Mining*, pages 261–270, New York, NY, USA, 2010. ACM.

[114] F. Wu and D. Weld. Autonomously Semantifying Wikipedia. In *Proceedings of the sixteenth ACM Conference on Conference on information and knowledge management*, pages 41–50, New York, NY, USA, 2007. ACM.

[115] L. Yang, T. Sun, M. Zhang, and Q. Mei. We Know What@ You# Tag: Does the Dual Role Affect Hashtag Adoption? In *Proceedings of the 21st International Conference on World Wide Web*, pages 261–270, New York, NY, USA, 2012. ACM.

[116] E. Zangerle and W. Gassler. Dealing with Structure Heterogeneity in Semantic Collaborative Environments. In S. Brüggemann and C. d'Amato, editors, *Collaboration and the Semantic Web: Social Networks, Knowledge Networks and Knowledge Resources*, pages 1–20. IGI Publishers, Hershey, Pennsylvania (USA), 2012.

[117] E. Zangerle, W. Gassler, and G. Specht. Recommending Structure in Collaborative Semistructured Information Systems. In *RecSys '10: Proceedings of the third ACM Conference on Recommender Systems*, pages 141–145, New York, NY, USA, 2010. ACM.

[118] E. Zangerle, W. Gassler, and G. Specht. Recommending #-tags in Twitter. In *Proceedings of the Workshop on Semantic Adaptive Social Web 2011 in connection with the 19th International Conference on User Modeling, Adaptation and Personalization, UMAP 2011*, pages 67–78, Gerona, Spain, 2011. CEUR-WS.org, ISSN 1613-0073, Vol. 730, available online at `http://ceur-ws.org/Vol-730/paper7.pdf`, accessed 2012-10-24, urn:nbn:de:0074-730-4.

[119] E. Zangerle, W. Gassler, and G. Specht. Using Tag Recommendations to Homogenize Folksonomies in Microblogging Environments. In *Social Informatics*, volume 6984 of *Lecture Notes in Computer Science*, pages 113–126. Springer, Berlin, Heidelberg, New York, 2011.

[120] J. Zobel and A. Moffat. Exploring the Similarity Space. In *ACM SIGIR Forum*, volume 32, pages 18–34, New York, NY, USA, 1998. ACM.

Curriculum Vitae

Dipl.-Ing. Eva Zangerle
`eva.zangerle@uibk.ac.at`
`http://www.evazangerle.at`

Education

- **PhD Studies** at the University of Innsbruck, Institute of Computer Science in the field of Databases and Information Systems (since 09/2007).
- **Master Studies** at the University of Innsbruck, Institute of Computer Science. Master Studies of Computer Science (04/2006–09/2007).
- **Bachelor Studies** at the University of Innsbruck, Institute of Computer Science (09/2002–04/2006).
- **Commercial College** (Handelsakademie), Landeck (09/1997–06/2002).

Work Experience

- **Research and Teaching Assistant**, University of Innsbruck, Institute of Computer Science, Databases and Information Systems Research Group (since 2007).
- **Proof-Reader**, self-employed, Innsbruck (since 2001).
- **Student Assistant**, University of Innsbruck, Institute of Computer Science, Databases and Information Systems Research Group (03/2007–06/2007).

Teaching Experience

- Einführung in die Informatik (Tutorial), winter term 07/08.
- New Database Models (Tutorial), summer term 2008.
- Bachelor Seminar (Tutorial), summer term 2008.
- Einführung in die Informatik (Tutorial, Coordination), winter term 08/09.
- New Database Models (Lecture and Tutorial), summer term 2009.
- Einführung in die Informatik (Tutorial), winter term 09/10.
- Information Systems (Lecture), summer term 2010.
- Einführung in die Informatik (Tutorial), winter term 10/11.
- Information Systems (Lecture), summer term 2011.
- Data Warehouse Systems (Tutorial), winter term 11/12.
- Information Retrieval (Lecture, Tutorial), winter term 12/13.

Supervised Bachelor Theses

- Konzeption und Implementierung eines Übungs-Management-Tools, Martin Bürgler, Clemens Müller, 2008.
- Entwicklung und Implementierung eines Wikipedia Annotation Layers, Rainer Frick, 2009.
- Weiterentwicklung und Optimierung der Applikation u2l, Philipp Vallant, 2009.
- Entwicklung des SQL-Tutors, Lukas Schwaiger, 2010.
- Weiterentwicklung und Optimierung der Applikation u2l, Florian Stäuble, 2010.
- Entwicklung einer Web-Oberfläche für SnoopyDB, Georg Unterthurner, 2010.
- Extraktion von semi-strukturierten Daten aus dem Web, Julien Poissonnier, 2011.
- Social-Media Konzepte in Recommendation-unterstützten Informationssystemen, David Hoppe, 2012.
- Entwicklung eines Frontends für das Hash-o-Mender Projekt, Benjamin Murauer, 2012.
- Entwicklung eines leichtgewichtigen HTML5 Web-Client zur Umsetzung von Social-Media Konzepten in nformationssystemen, Thomas Fraydenegg, 2012.
- Entwicklung eines leichtgewichtigen Web-Clients für ein Hashtag-basiertes PIM-System, Robert Bierbauer, 2012.
- Entwicklung eines selbstlernenden Vorschlagmoduls für ein Hashtag-basiertes PIM-System, Markus Müller, 2012.
- Entwicklung des freien Lernsystems Kakadu, Alex Lanz, 2012.
- Entwicklung eines leichtgewichtigen Web-Clients für das Lernsystem Kakadu, Georg Schmidhammer, 2012.
- FaceFinder: Gesichtserkennung in OwnCloud, Aaron Messner, 2012.

Supervised Master Theses

- Reasoning with Ontologies in Databases and Optimization Strategies, Doris Silbernagl, 2009.
- A SnoopyDB Prototype, Michael Tschuggnall, 2010.
- Exploring Structures of Infoboxes in Wikipedia, Alexander Larcher, 2010.
- Recommendation of Structured Tags, summer term 2010, Martin Bürgler, 2010.

- Fast and Scalable Recommendation Computation in Semistructured Collaborative Information Systems, Tim Hannemann, 2010.

Awards and Grants

- Research Scholarship of the University of Innsbruck (Nachwuchsförderung der Universität Innsbruck, 11/2011–11/2012.
- Tyrolean Science Fund (Tiroler Wissenschaftsfonds), 12/2012–11/2013.
- Österreichische Forschungsförderungsgesellschaft FFG (Innovationsscheck together with FIOconsult e.U.), 10/2012–10/2013.

Professional Memberships

- Deutsche Gesellschaft für Informatik
- Association for Computing Machinery

Professional Activities

- Member of PC, 9th International Conference on Semantic Systems (I-Semantics), Graz, 2013.
- Member of PC, 8th International Conference on Semantic Systems (I-Semantics), Graz, 2012.
- Head of Organization Commitee, Grundlagen von Datenbanken Workshop 2011, GI Fachgruppe für Datenbanken.

Other Activities

- FIT—Frauen in die Technik, Austrian initiative aiming at attracting girls for technical studies, various talks about studying computer science, short tutorials.
- Admina.at, Hardware Tutorials for female students, University of Innsbruck.
- Girl's day, Tyrolean programm aiming at attracting young girls to technical professions, teaching basic programming skills to girls aged 12–16, University of Innsbruck.
- Infoküche Podcast, Regular contributor to the Infoküche Podcast concerned with Information Systems, `http://www.infokueche.at`)

Book

- Stefan Pröll, Eva Zangerle, Wolfgang Gassler. *MySQL: Das Handbuch für Administratoren*, Galileo Computing, Bonn, Germany, 2011, `http://www.mysqladmin.at`.