Hermann Ziak, BSc

Context Driven Federated Recommender Systems in Uncooperative Settings

Master's Thesis

Graz University of Technology

Institute of Interactive Systems and Data Science
Head: Univ.-Prof. Dipl-Ing. Dr.techn. Stefanie Lindstaedt

Supervisor: Dipl-Ing. Dr.techn. Roman Kern

Graz, Jan 2018

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, _____ _____

Date Signature

Eidesstattliche Erklärung[1]

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am _____ _____

Datum Unterschrift

[1] Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Acknowledgement

I would first like to thank my advisor Dipl-Ing. Dr.techn. Roman Kern for his continuous support, effort and constructive feedback over the recent years.

I would also like to thank my family, especially my parents, and friends for their support throughout my years of study.

Finally, I must express my very profound gratitude to my girlfriend for providing me with unfailing support and continuous encouragement.

Hermann Ziak

Graz, January 2018

Abstract

Today's search engines are tailored towards the information need of the average user. This leads to the problem that valuable sources of information that are highly specialized towards a subject can be underrepresented since in general search engines optimize towards returning most popular results. One possible solution to approach this problem is the development of a system that allows integrating such sources in one single framework; typically called a federated search engine. Since just returning all the information from all sources back to the user might not full-fill the user's specific information need or reflect the subject-specific knowledge of the user personalisation is required. This could be achieved by adapting algorithms from the field of recommender engines. Ideally, such a system should detect the users need of support from the context without having her explicitly taking actions.
Based on the initial question, how to create such a system that automatically detects the information need of its user and returns personalised result lists, four research questions were formalised. First, it was necessary to identify the needed means to extract topics out of the textual context.
Second, the challenges arising from automatically generated queries needed to be identified, and potential methods to overcome those challenges. Additionally, which query processing steps are beneficial in this setting?
Third, if and how the collection representation can be further optimised especially in volatile uncooperative settings.
Fourth, how distributed document retrieval can be personalised, in particular when highly precise results are of lesser importance. Furthermore, which aggregation techniques are beneficial in this setting.
Based upon these questions a total of six experiments were conducted resulting in six publications in conferences and workshops. The results of this work shows that the framework of a federated search engine can be altered towards a federated recommender system that allows recommending items

to users automatically; tailored towards the user's information need and context.

Contents

1 Introduction

Recommender systems and search solutions are nowadays common tools for all kinds of online platforms to raise attractiveness to their user and increase revenue. Since the volume of data has increased dramatically in recent years the traditional system architecture of one single instance that handles all request is not always feasible anymore. Even further it's not always reasonable to replicate all data that could be helpful for the underlying task. Thus, federated systems in these areas are more frequently needed.

The common goal of recommender systems and search machines is to present items or information to the user which is of her potential interest [15]. Whereby both often try to guide the user in a personalised fashion towards them. Often these systems analyses either the content of the providers, for example, the products in on-line stores, or the data produced by the users, e.g. which products are typically bought together or help the user by filtering the information [62]. Self-evidently these systems are not limited to only these approaches, and a vast amount of different approaches has been developed over time.
In general, a basic resemblance between the goals of recommender systems and search engines can be seen. Although, the main purpose for a search engine, on the most basic level, is to find relevant information towards a search query from a vast amount of information. Typically, a traditional search engine asks the user to state his information need. Therefore the user has to trigger the system to produce results intentionally. Based on this information some kind of similarity measure is applied to the user's query and the items where as all the matching results within the collection are rank and the top elements returned; an information retrieval task. In contrast, many recommender systems are triggered implicitly by the users' action or history. For example by selecting an item of the user's interest from the collection and matching similar items towards it; an information

filtering task. But again an emerging subfield of search tries to integrate this automatic triggering of the system into the field information retrieval as well. This process of pro-actively reacting upon the users need is called Just-In-Time Information Retrieval [58].
One other typical characteristics of Recommender Systems is the usage of other implicit information like information that is obtained by analysing similarities between users, information that is provided by the provider or the users' history. But looking at today's major search engine making use of such information to improve results and better cover the users intend it is no unique characteristic to recommender engines anymore. For example, the integration of information from different sources like news or images into typical text-based results is often based on typical user behaviour which could be seen as recommendation approach as well. These examples show that the boundaries between search and recommendation seemed to dissolve. At least one could find that search engines are often improved by incorporating techniques from the field of recommender engines. Still, automatically retrieving items purely through the users' action and context is not applied often.
The biggest dissimilarities of this two fields could potentially be found in the main problems which they are facing. Recommender systems typically try to resolve problems like data sparsity, the cold start problem and typically handle vast but often well-described data.
Search engines, on the other hand, generally try to resolve problems like processing of natural language, efficient extracting and indexing of data and effective ranking of result sets typically measured in precision and recall. The federated setting even leads to further challenges like collection representation and selection or result merging. Some of these can be resolved with techniques of the field of cross vertical aggregated search. Which allows integrating different types of sources into one set of for example textual results (e.g news, videos). Although it is a common feature in most major search engines today [41], this technique does not necessarily resolve all issues arising. Through the mass of content available today often only the most popular or prominent content reaches the users which might not cover the information need in the full extend [32]. There is a huge amount of different pieces of information that might be too specialized for some users but would be beneficial for others. Therefore, to promote more specialized content of potentially helpful niche sources, systems that are

capable of retrieving these items can be applied, for example Federated Search Engines [17, 71]. Therefore Federated Recommendation Engines could be implemented as a synergy of both, using approaches of both fields; recommender engines and federated search.
One of the challenge in fusing these two fields is the automatic query and user profile generation. The final automated query has to cover several aspects of the users' information need and context. The information need can not only be extracted from the user's context but should, at least to a certain degree, incorporates the user's history. Further, it could be used to enrich the user's profile. For example the level of expertise of the user in a certain field. This can be done either by relying on specific terms in the query itself or certain indicators in the according user profile [58]. Today major search engines track the users' behaviour over time and generate the user profile based on the stored information. But from a users point of view, this action can be seen as an invasion of the users' privacy. An alternative could be a client side solution that leaves ma user in charge what he wants to share with the system.
This thesis covers an overview of recommender system approaches, federated and aggregated search, a brief description of a system that incorporates ideas of the field of federated search uncooperative setting and recommender systems and finally experiments that were conducted.

The main question is how to create a system that automatically responds in a personalised way to an information need of the user while relying on a vast amount of diverse collections of knowledge. This main question lead to the following research questions this work aims to answer:

Context Detection Which means are necessary to extract topics out of the textual context?

Query Processing What challenges arise from automatically generated queries and what are potential methods to master those? Additionally, which query processing steps are beneficial in this setting?

Collection Representation Can collection representation be further optimised especially for uncooperative settings?

Personalized Result Aggregation How can distributed document retrieval be personalised, in particular when highly precise results are of lesser

importance? Furthermore, which aggregation techniques are beneficial in this setting?

In the attempt of answering this questions several experiments were conducted which resulted in six publications:

a) "KNOW at the Social Book Search Lab 2016 Mining Track" [85]
Hermann Ziak, Andi Rexha, Roman Kern
The paper describes ans approach for automatic assistance request detection of users in unstructured text.
Personal Contribution: System design, feature engineering, data processing and execution of the experiments. Scientific supervision by Roman Kern.

b) "Query Splitting for Context-Driven Federated Recommendations" [83]
Hermann Ziak, Roman Kern
The presented system described an approach to split continuous queries covering various topics through the usage of word embeddings.
Personal Contribution: Experimental design, feature engineering, dataset generation, processing and execution of the experiments. Scientific supervision by Roman Kern.

c) "Do Ambiguous Words Improve Probing for Federated Search" [76]
Günter Urak, **Hermann Ziak**, Roman Kern
This work describes a novel knowledge based approach for source selection in federated search with the additional goal of promoting highly specialized collections. Further, it describes an improvement for query set generation in query based probing. Personal Contribution: Dataset preparation and general supervision and assistance. Scientific supervision by Roman Kern.

d) "Efficient Search Result Diversification via Query Expansion Using Knowledge Bases" [61]
Raoul Rubien, **Hermann Ziak**, Roman Kern
The paper describes an approach to diversify result sets efficiently by altering the input query via pseudo relevance feedback. The proposed

approach is evaluated against state of the art approaches.
Personal Contribution: Dataset preparation and general supervision and assistance. Scientific supervision by Roman Kern.

e) "Evaluation of Pseudo Relevance Feedback Techniques for Cross Vertical Aggregated Search" [82]
Hermann Ziak, Roman Kern
This work describes the design and setup of a framework to conduct a user based evaluation for cross vertical aggregated search. Furthermore, it presents results of a conducted evaluation in this setting including a general guideline for such evaluations
Personal Contribution: System design, data processing and execution of the experiments. Scientific supervision by Roman Kern.

f) "Evaluation of Contextualization and Diversification Approaches in Aggregated Search" [84]
Hermann Ziak, Roman Kern
This paper describes an evaluation conducted on a crowd sourcing platform to compare block ranking and interleaving techniques in the setting of personalized search.
Personal Contribution: System design, data processing and generation as execution of the experiments. Scientific supervision by Roman Kern.

2 Related Work

2.1 Recommender Systems

In the following section an overview of common recommendation techniques will be given. Due to the great diversity of approaches within this area only the most prominent and relevant will be briefly addressed in this section. Although, some of these approaches might not be applicable in a federated setting where the data of the collections, and therefore the set of recommendable items, might not be as consistent as usual. A federate recommendation engine might not even have access to all possible items within the source and might even have to rely on a limited amount of ad hoc retrieved content. Recommender systems often use collected information of their users, acquired either explicitly, by users stating their preferences for example on a rating from one to five like a Likert scale, or implicitly, inferred by their interaction with the system. Aside from these kinds of data, that can be used by applying statistical methods and similarity measures, different kinds of other information can be exploited as well. For example; demographic, social, item based or context based information [30].
There are several possibilities how a user can express her information need which represents the desire or need of a certain information. I) Through a query, a user might be able to give precise information about the expected specifications of the item of interest. For example the expected amount of mega-pixel of a digital camera. II) By giving certain constraints the user can further narrow down a number of items. Sticking to the camera example, this could be perhaps the maximum amount of money the customer wants to spend. III) The customer could give further preferences that might not be hard constraints. For example, if she prefers to stick to a particular brand of camera manufacturers if another offer is not significantly better. IV) A fourth source of information that a recommender engine might want to take

into account is the context of the user. For example in the case where a user intends to purchase the item in a camera supply store close to her location, the system should rank the best offers nearby.

To allow the recommendation engine to filter the results a lot of content knowledge is needed of the items which can be categorised into groups.

Attributes of the item Attributes of the item like the price or weight can be applied as filter.

Contextual information of the item Similar like the location of the user can be used to recommend items in a particular proximity contextual information can be utilized as well.

Domain knowledge To a certain extend knowledge of the given domain of the products recommended can be used.

2.1.1 Collaborative recommendation

Aside from our own preferences and experiences, humans tend to be influenced by the opinion and knowledge of others to make decisions. Collaborative Filtering (CF) is one approach to mimic this typical human behaviour. The concept of homophily, the general tendency of people to share similar characteristics in groups, is utilised to group people in the system by the expressed interests of the users. For example, the explicit or implicit rating of the users for certain elements (e.g. books, movies, songs) can be used to calculate a similarity score. By relying on this similarities, items or users can be clustered. The knowledge gathered from the resulting similar users within the cluster can then be used to recommend new items to the current user. Today this similarity is often calculated by making use of the k Nearest Neighbours (kNN) algorithm; in earlier times measures like the Pearson correlation or cosine similarity were proposed as well [39].
The collaborative recommendation approach can be further separated into two subgroups. I) User-Based: Here users with similar preferences are selected and highly rated items within this set of users are recommended

to the initial user. II) Item-Based: Whereas in contrast to the User-Based approach the similarity is calculated between all items and the currently presented item.

Cold Start Problem

On of the biggest challenges of the CF approach is the so-called cold-start problem. For a newly introduced system, or when introducing a new user, the rating matrix is going to be empty or sparse. This implies that it is not directly possible to apply similarity measures between items or users with decent results. According to Bobadilla et al. [11] the general cold-start problem can be further partitions into three subproblems. I) The "new community" problem, where after starting up the new system no ratings are given, and the rating matrix is empty. II) The "new item" problem refers to the situation when a new item is introduced into the recommendation engine, and no ratings are available for that particular item. III) The "new user" problem where a new user is registered in the system and no information about her preferences is available by that time. When a collaborative filtering based recommender system is initially started it will lack the needed data in the rating matrix to calculate similarities between users or items [11].
One possibility to resolve this problem faster is to design the system in a way where users are initially steered towards rating items. The new item problem, where a newly introduced item is unrated, leads to the issue that the users might not get aware of it since it is never recommended. This can lead to the point where an entire section of the collection is never noticed by the users. One option to resolve this issue is that newly introduced items are presented to a particular user group who are willing to rate these items. To resolve this issues, several approaches were proposed in literature. For example, one option, presented by Brees et al [13] proposes a default value for items where only one user has stated his preference. Another algorithm by Huang et al [37] tackles this issue by the transformation of the rating matrix into a bipartite graph where within this graph an exploration algorithm is used for the analysis. A further possibility to circumvent this problem was presented by Wang et al [78]. Here a hybrid of user and item based similarities is used to improve the prediction accuracy.

2.1.2 Content-based recommendation

Collaborative filtering techniques rely on ratings from users which have to be gathered by the system in production. In contrast a content-based recommendation approach can use data that is acquired beforehand and does not suffer from the cold start problem. Categorical data of movies or similar like genre, actors or directors can be gathered and used to recommend those to users with similar interested stated in their profile. For example, if a user states her interests in several science fiction movies, there is a high probability satisfying recommendations should contain movies in this genre as well. Although the preparation and maintenance of such meta-data is a lot of effort, it has the benefit that the new item problem does not arise. Though, one problematic issue is that the usefulness of the meta-data depends on the underlying document set. The given example with movies will have another set of the main features than, for instance, plain text documents, books or music. In the simplest setting, a content-based recommender engine maintains one list of these features that are assigned both to the users, e.g. the interests, and the items. These interests of the user have not necessarily to be stated by the user explicitly. They can also be implicitly gathered by letting the user select items she has already consumed. For example, combined with an indication of how much they liked the item. The recommended item set is then generated through pairing the user's interests with the information available from the items.
In this setting recommender system might reuse ideas from information retrieval to improve the restrictive information given only by features like the author.
One example could be to use a list of keywords extracted from purely textual items. This could be used with a very simple boolean model. (e.g. The searched keyword is either in the document, or it is not, and therefore the item is either relevant or not relevant. Therefore a ranking of the items is not given.) Though, there are several drawbacks to the simple boolean approach. First, it does not take the length of the document into account which would lead to the promotion of longer documents. Second, each term is considered equally relevant what is typically not the case in natural language. Another more sophisticated approach is making use of the vector space model which is usually used in search engines. This is often achieved

by the use of the TF-IDF measure. It uses the term frequency of the query term within the documents. This frequency is later multiplied by the inverse document frequency. The inverse document frequency represents the inverse occurrence of the term in the corpus. The final result is the relevance of the term in relation to the corpus. To produce then needed information for this task at indexing time each term in the documents is analysed. The resulting data structure can be seen as a vector space, where the size of the dimension aligns with the number of keywords in the documents [59]. Alternatively similarities measures can be applied to obtain sets of similar items, similar to the collaborative filtering approaches. For example, the kNN method can be used, which is relatively simple to implement. Further improvements can be made by focusing on recently rated items as short-term recommendation set and the rest to cover long term interests. That way the system can comprise the current interests of the user more suitable and also reduced computational effort. To combine long and short term predictions, one could consider using long-term predictions if there are not enough similar items retrieved through the short term prediction step. Alternatively, it would be possible to calculate the short-term prediction in an on-line fashion. The computational expensive long term predictions could be calculated when system loads are low and processing power is not an obstacle.

2.1.3 Knowledge-based recommendation

Although collaborative filtering and content-based recommendations have shown to be effective in many applications and are widely used they both have their weaknesses in certain situations. The most prominent arise in situations when irregular purchased items should be selected. Relying in this case on the item or user similarities might not produce satisfactory results. Furthermore relying on users ratings for such item would have the drawback that they might get obsolete over time and the collected data might already be outdated. With today's pace of technology, a good rated two year old Smart-phones will not have features consumers today would expect in modern technology. Such items, for example, could be Smart-phones, TVs or Computers. In such situations, knowledge-based approaches can be used. Distinguishing feature from collaborative filtering and content-based

recommendations is that knowledge-based recommender engines tend to have a highly interactive process of narrowing down the potential results to a subset of interesting items. Therefore they are considered as "conversational systems" [15]. Typically the process to generate interesting items for the user is too narrow down the result set by defining requirements. If no results fit the given restrictions, the user has to reduce this requirement. A step that can be supported by the system through recommending items which do not meet all the restrictions or suggestions for relaxation of the constrains [39]. In this field different basic types can be distinguished, constraint-based [30] and case-based [14] , query based [40], ontology based [48] or social knowledge based [21].

Constrains-based

Constrains-based recommendation approaches can be applied in situations where highly specific requirement have to be met. [30] This approach requires a quite exact definition of the user's preferences and an extensive knowledge base in regards to the items. Though, a so-called constraints network is built upon the constraints and two sets of variables [30]. One set of variables can be described as the customer's properties (U) the other one describes the features of the product (P). The constraints group again consists of three subgroups; product, filter and compatibility constraints. Compatibility constraints limit the item set through factors that have to be met to suit the user's application. For example, if the user needs a camera for taking pictures at sports events a fast auto-focus will be required.

Case-based

In a case-based recommender system, the main goal is to find similar items towards the item the user has in mind. In this case, the similarity will often be measured by the use of external knowledge, in particular, domain-specific knowledge. The whole process is based on the case-based reasoning methodology. Here a problem is approached through looking back at already resolved similar problems. The already known solutions to the problems are used to approach the current problem. These problems and

their solutions together determine the cases in case-based recommenders. In its most basic form, a case and an item are identical. To gather the recommendations the user states her preferences. Upon this preferences and the item set, a similarity measure is applied to retrieve a set of products which are presented to the user. Typically these recommendations include an explanation and allows the user, by the help of the user interface, to alter the stated requirements [44].

Query-Based

A constraints based solution can also rely on conjunction queries in databases [39]. Here the item selection problem can be seen as filtering task. For example, a query can be constructed that contains information like the price, the brand and other features of a product that conjunct in one SQL statement which returns only items that fall into this constraints. These constraints do not have to be formulated directly by the user but can also be derived from auxiliary information given. Jannach et al. [39] gives the example of a camera for the purpose of printing posters. This implies that the resolution of the item should at least exceed a certain minimum value.

2.1.4 Context

To only take users and items relationships into account is often not sufficient in many applications. For travelling, recommendations further contextual information could improve the recommendations significantly. For example, the location of the user could be substantially relevant in the context of sightseeing or the season of the year could influence the location of the trip significantly. These two examples already give a small insight of the broad spectrum of possible context that can be taken into account. Though, in literature context awareness for recommender systems has no unambiguous definition. Some definition only takes into account the users surrounding like in ubiquitous computing. Other definitions include almost any information in respect to the underlying domain [39]. Shilit et al. [64] defined to be the most important context of the users in recommender systems with the three questions. Who you are with, where and what is your surrounding.

A more extensive definition of the user's context categories is given by Ranganathan et al. [57]:

- physical contexts (e.g. location, time)
- environmental contexts (e.g. weather, light)
- informational context (e.g. sport scores)
- personal context (e.g. mood, activity)
- social context (e.g. group activities)
- application contexts (e.g. email, visited websites)
- system contexts (e.g network traffic)

Typically these context related pieces of information are accumulated in a user model, or user profile, that is used in the recommendation process. Never the less this information has to be handled with care. On one hand, the usefulness of contextual information can change over time, and this timespan might even be brief. A user that is searching for presents for the whole family could go directly from children's books to historical novels. On the other hand not every contextual information is relevant in every situation. A comprehensive discussion of this topic is given by Dourish et al. [28].

2.2 Federated Search

Classical information retrieval typically model search under the precondition of one centralised index [4], or at least a distributed index with combined statistical information about the content of the indexed collection. The alternative, to search in a distributed manner, to amalgamate different data sources, a technique called federated search or distributed information retrieval can be applied [7]. Figure 2.1 show the typical architecture of a federated search engine with its components. Three major steps can be outlined in a typical federated search engine [66]; First queries are sent from one client to the federated search engine or broker. Often this system then selects based on the content of the query and other information a subset of the available collections. For example, the queries can be categorised by the users intend or potential domain [72]. In the next step, it transforms the request into a query that the selected source can process and triggers a

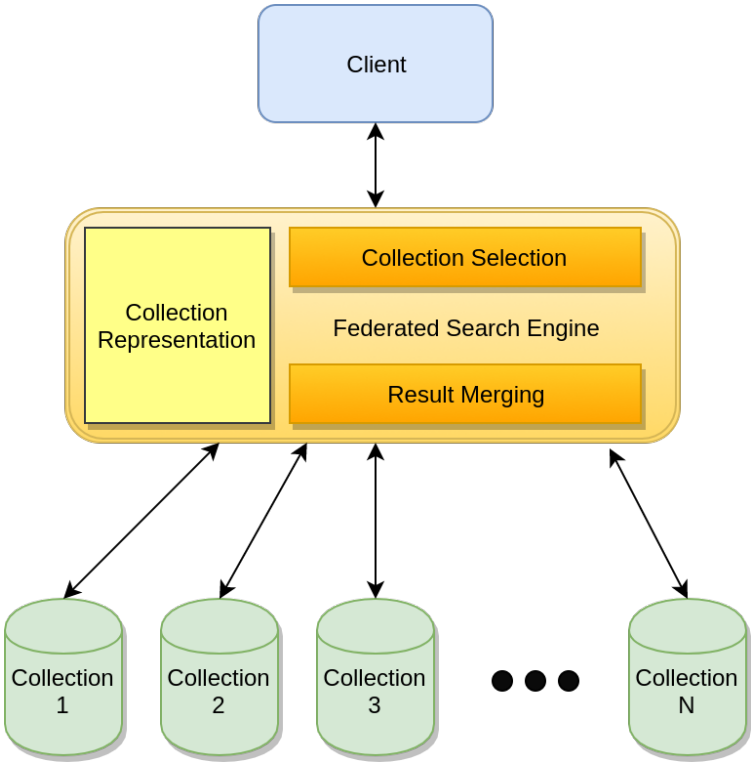

Figure 2.1: This figure shows the typical main components of a federated search engine. The "Collection Selection" which has the task to reduce the amount of collections registered to the system to a smaller set. This module makes use of the "Collection Representation" module which has the purpose to gather all the needed information of the connected collections for the collection selection process and store this information. And finally the result merging component processes all the received results from the collections to generated one unified response to the client.

search request. Finally, returned result have to be converted into a unified item list and based on this list a final result list is composed. This process of composing the result list involves the identification of best matching items and posturing of different kinds of verticals. These so-called verticals are describing type-sets of retrieved items from a search engine like text, pictures, videos, news. In contrast to traditional retrieval, since the statistical information is already processed by the collection, little information about the collections is needed to retrieve items. Typical traditional search engines have an own component, the crawler, which extracts the data from web pages to be indexed by regularly revisiting the pages to monitor changes. This process includes information that is already indexed by specialised search engines. In federated search already indexed data does not have to be duplicated to retrieve items and a federated search engine is not in need of monitoring minor changes within the sources collection. In general, a federated search engine has no need to store or index documents at all. This allows even to integrate systems that can be categorised as uncooperative. In this setting, an uncooperative source is one that does not provide statistics or meta-data that describes their contents by them self. Though, for the selection of these collections, the federated search system is still in need of some sort of statistical information. Therefore, if the information is not provided by the API, there has to be a mechanism in the broker that gathers data which represents the collection's content of the source. Federated search also has some other benefits compared to the traditional search model. First, indexing the hidden web, e.g. the not directly accessible content of a search engine, is a hard task for search engines [56]. Second, the crawling process is expensive regarding processing and storage, especially with today's rate of changing content on the web [66]. Still, the process of federated search, as already described, does not come without its own set of challenges. Thus, literature discusses three main challenges [67, 45]:

collection selection The process of matching the query or user profile to a source.

collection representation The task of getting uniform statistical and lexical information of each source in the broker.

result merging The incorporation of the retrieved items into one single result list that can be presented to the user.

2.2.1 Collection Selection

The first step in federated search is collection or source selection. Here the broker has to decide which collections fit the query or user profile best. This step is mainly important for two reasons. Not only does it allow to provide more relevant content to the user. Even further, with the precondition that statistical and meta-data data of the source is given, it allows to reduce the traffic passing through the system and reduces the load. This allows the system to adapt to temporary performance issues as well. If one of the collections is down for maintenance or faces high-traffic, the broker can take this into account and might down rank the importance of the source for a period of time. Since in general the performance of the federated search engine is determined not only by its limitation of processing the requests but mainly by the response time of the underlying collections. Basically, it can not respond faster to a request than the slowest source selected for the query. Through recent years several different approaches where suggested. For example, lexicon based, document-surrogate based or classification based [67], which will be presented in the following paragraphs.

Lexicon Based

In lexicon based collection selection approaches lexical statistics of the collections are generated or retrieved to compile a representation of their language model. This representation can be used to rank the collections according to the query. When these kinds of statistics are not given by the collection directly, they can be created by sampling. This approach found a lot of applications and is widely researched [67]. Further, it has the benefit to work with cooperative and uncooperative collections. In recent decades different approaches have been proposed. For example, like the bGLOSS method [43], which relies solely on term frequencies, or later vGLOSS which uses a Cosine similarity of document vectors and the query [33]. Finally, the similarities are summed up to calculate the usefulness of the source towards the query. Another example could be the collection retrieval inference network or CORI net [16]. In this inference network the leaves represent the collections and the nodes represent the terms in the collection.

The needed probabilities for the edges can be calculated analogue to the term frequency (tf) and inverse document frequency (idf).

Document-Surrogate Based

Document-Surrogate methods try to rank queries to a collection but also take information of the single documents into account [67]. One of the most prominent examples is the ReDDE algorithm [70]. This method uses the content similarity in the source selection process but as well tries to estimate the database size through sampling. So in comparison to CORI, through taking the database size into account, the document ranking could be improved. Further, smaller highly specialised databases have a better chance to be selected. This approach estimates the size of the collections through the sample-resample method. In this method, a query from the resource description is created and send to the collection. Typically retrieval systems return the full amount of potential matches within the response. This amount is used to estimate the potential size of the collection. Additional one-term queries are sent to the database to approximate the size further. These methods can be applied when the collection gives the potential query terms through a resources description (e.g. the most significant unique terms of the collection), but it is also applicable when such information is not available. A similar approach is followed in the centralized-rank collection selection method (CRCS) [66]. The main difference to ReDDE is the integration of a further source of information; the actual rank of the document in the retrieved list from a centralised index. In their work they propose two possibilities to calculate the actual weight. One is a linear descending weight according to the position in the list. Since, as stated in their work, the importance of documents to users have a negative exponential relation the second is calculated as exponential descending weight. The authors report that, depending on the used collection, CRCS outperforms ReDDE in most cases in its precision, especially when no linear weights are calculated.

Classification Based

According to Arguello et al. [4] three main group of features can be identified to classify collections.

i Features based on categorization of the query.
ii Features based on the corpus or metadata of the corpus. Extracted for example by the use of methods like ReDDE.
iii Features based on click-through information captured by the system through monitoring the user's behaviour.

A system relying on the first group is described by Centitas et al. [22] in the qSim method. It is based on the assumption that there is a tendency of many similar queries in federated search [22]. Though; classification could be used to learn similarities to training queries. In this approach, the subset of collections is selected through comparing past queries to the current query. The collections are ranked through their average performance for past similar queries. To calculate the performance of the collections a the ranking of documents in a centralised index is used.

Arguello et al. [4] proposed an approach which applies a logistic regression model to the problem. In an initial step, it makes use of all proposed sets of features to produce binary predictions for each collection. They concluded that their multi-stage classification approach either performed at a similar level or significantly better than others like CORI or ReDDE.

2.2.2 Collection Representation

Traditional search engines rely on statistics of the indexed corpora. Through this very process, a statistical relationship between the query and the indexed documents can be calculated. In federated search these statistics are typically not accessible, therefore a comparison between documents and sets of documents from the connected collection is more challenging. In a cooperative environment meta-data and further information would be given by the system itself. Some algorithms like CORI or gGloss rely on this assumption. Unfortunately, it is typically not the case [67] that public APIs of search engines provide this information. In such an uncooperative

setting the needed data has to be gathered by the system itself to support the collection selection process. This also allows overcoming the potential risks off provided data from search engines on the web. (Like the lack of incentive of the content provider, the potential misrepresentation of the content to achieve higher rankings [8, 18] or the potential variety of data provided by such services [18].) A further issue is the estimation of the size of the collection in question. Some collections might only contain a small number of documents which are still highly relevant for certain queries. Bigger collections, depending on the selection algorithm, could dominate these small collections by their sheer size. Therefore, this information can be highly valuable for the selection algorithm. Unfortunately, not only is the information not available directly, many public APIs return only a certain maximum of documents to a particular query [8].
In general, it can be assumed that the aggregation of comprehensive information of the underlying collections before a source is integrated into the selection process is beneficial. Though, earlier algorithms were proposed retrieving information ad-hoc through sending an initial query towards each source shortly before the actual query is sent to calculate a ranking based on this initial documents. However, in a setting where a vast number of collections is present this approach leads potentially to a far longer response time. Even further, potentially valuable information about the size and content of the collection can hardly be estimated this way. The most prominent approach for creating a representation of the content of the collection is query based sampling [19].

Query Based Sampling

Query based sampling was initially introduced by Callan et al. [19] for searchable text databases. The process is based on the assumption that it is sufficient to send a certain amount of queries to a search engine to create, out of the resulting documents, a description of the content of the collection similar to a description created with full access. Callan and Connel [18] state in their evaluation of the approach that the amount of queries and documents needed is an open question although they suggest about 300-500 unique documents. This suggestion was rebutted by Shokouhi et al. [69]. They conclude that an approach relying on a number of new unique terms

within the result set is more reliable, especially with bigger collections, instead of a fixed amount of documents. For the set of queries, there is still no optimal solution. Although, in literature, it is a widely accepted method to draw random queries for this purpose [69]. Since just sending random queries to the search engine might not return uniform document sets Bar-Yossef and Gurevich [8] presented a graph-based solution that should potentially resolve this issue. The graph is built upon the documents and their connection through the query. If a query returns the same documents, which represent the nodes, they are connected by an edge. Upon this graph, the algorithm performs a random walk. When a node matches a new query this query is sent to the search engine. Afterwards, out of the new result set a new document is selected. Another approach to mainly estimate the size and overlap of search engines was introduced by Bharat and Broder [10] and can be used for random sampling of documents from a search engine. Through a preselected keyword set of an existing corpus, sorted by their frequency, the algorithm constructs multi-term conjunction and disjunction queries. The conjunction queries are constructed out of two query terms and selected to return between 1 and 100 results. The disjunction queries are constructed out of four query terms. From the results of the send queries, a random result is picked. By this randomly selected results, they estimate the size of two search engines through their intersection. Given that $1/2$ of the samples of one search engine is also present in the second search engine and $1/6$ of the samples in the second search engine is present in the first search engine one could estimate that the second search engine is about 3 times bigger in size than the first. Although, this approach can only be used for size estimation if it can be expected that the indexed corpus is supposed to contain the same content the approach could be still applied to gather a uniform result set. In 2007 Thomas and Hawking improved this technique further [74] to achieve better estimations.

2.2.3 Result Merging

One of the main challenges in federated search is how the results of the different collections or verticals can be represented to the user to optimal suit the ”information need”. The simplest solution to present several collections

to one user would be representation in separated taps. It can be argued that presenting results from all available collections is beneficial for the users. A strong argument was brought forward by Sushmita et al.[72]. In their analysis of query result logs, they conclude that restricting the user's view to strongly filtered results might lead to withholding valuable information. Nevertheless, in a federated search setting where one can assume that there might be a high amount of verticals or collections it might not be feasible to present all collections.

According to Arguello et al.[3] over time three main concepts emerged to aggregate several results list or verticals in a combined result representation. Although these approaches are typically used purely for combining verticals in an aggregated search environment, their concepts can be transferred to federated search as well.

i A set of verticals is chosen which has a high probability to be related to the query.
Then compared with the main result list a preview of the top results for these verticals is presented in complimentary to the main result list. This technique can frequently be seen in web-search engines where it might be most suitable.

ii An alternative approach is blocking and interleaving. In the blocking method, the top results of each vertical are taken and stacked. Here it would be reasonable to rank the verticals beforehand according to their relevance. Alternatively, for instance, when all verticals are considered to be of similar importance, these block could be arranged in a perpendicular manner. Aside from the possibility of blocking results from different collections can be interleaved. In a simple implementation from each source, the top item is integrated into the final list item by item. Several different algorithms to improve interleaving where proposed over time. A comparative analysis based on click-through data of several interleaving methods was reported by Chuklin et al. [24]. Typically these approaches can be applied in the federated search setting as well [16]. According to Arguello et al. [3] a direct comparison between interleaving and blocking was not performed until today. Another open issue is still the weighting of the verticals to

determine the ranking from which vertical to pick items first.

iii Third, a so called aggregated search result page (SERP) [3] can be created. Typically the task of creating a SERP can be separated into two stages. First; the task of selecting the right vertical in relation to the query. For example results from verticals that can be considered irrelevant can be excluded from the final result list. Second; the process of deciding the sequence of the verticals in the final result list. Alternatively, if no vertical should or can be dismissed, there is also the option to present highest ranked collections results in predefined spots but still, present the user the full set of retrieved results.

Over the last decades, in the field of federated search, several systems were proposed to resolve the result merging problem. In CORI Callan et al. [16] try to calculate a new score of the document out of the retrieved document score. Here they rely on the score of the collection produced by the system and the document score from the collection. To achieve comparable values, the scores are normalised through a heuristic.

3 Context Driven Federated Recommender

A context driven federated recommender system can be seen as a powerful combination of a federated search engine and several techniques commonly used in recommender systems. Figure 3.1 shows an illustration of the potential architecture of such a system.
In the case where one would like to promote content that is typically under-represented in major search engines, like highly specialized culture or historic content, and simultaneously support user in creative processes like authoring documents to certain topics a federated recommendation approach appears to be promising. In such a system the recommendation approaches can be applied on multiple modules to improve the general performance and to tailor the results towards the users information need. The main components of a federated search engine mostly derive from their challenges described in 2.2. For example the source selection module can be seen as typical filtering task. Given that contextual information of the user is existing, one might want to filter potential collections by techniques used in knowledge-based filtering 2.1.3. Still there is a difference in the process. The given constrains are potentially not directly defined by the user but automatically inferred by the users actions or context and therefore can not be handled that strict. For example the selected sources can be limited to collection that contain only a certain type of meta-data. Another example can be the result aggregation module. Here the task is more a re-ranking of given items instead of a filtering task. In this case similarity measures like in content-based recommendation approaches, discussed in 2.1.2, could be applied.
To get an overview of the needed processes and modules in a federated recommendation approach the potential work-flow of such a system can

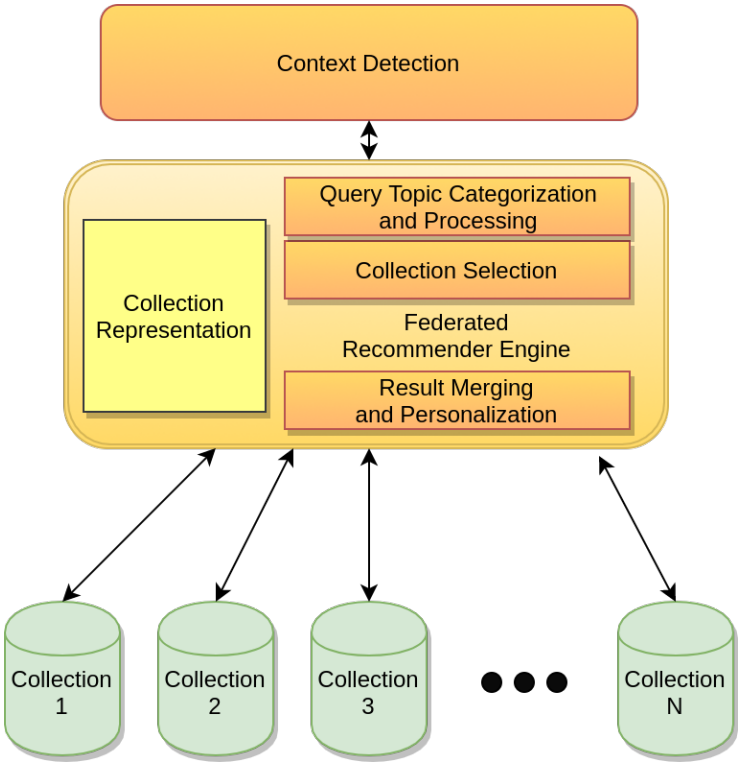

Figure 3.1: Similar to figure 2.1, the architecture of federated search, this figure shows the main components of a federated recommender engine. The simple client is replaced by an intelligent agent, the "Context Detection", which represents a separated entity to the system and runs on the client. Furthermore the "Collection Selection" is preceded by the "Query Topic Categorization and Processing" module. Finally, the results are personalised and merged in the last module using information from the preceding processes including the "Context Detection".

be used as basis. The first step in the work-flow of the system would be the identification of the information need of the user. The context of the user's current working environment could be used to generate the main topic on which the user is currently working on. Further the history of the user can be used as valuable additional context. For example, when a user plans to write an article about ancient Egypt, potentially a art student might have another underlying focus or main topic on the article then a student of architecture. Although the context used in this example is just the current information that is presented to the user and the short term users history but also the definition of context used in recommender systems, discussed in section 2.1.4, can be of value. For example even the location of the user might add further valuable information. When all the information is gathered it has to be analyzed to identify the actual task of the user in which the system should support her. Creating the results just purely on the gathered information might not lead to adequate item recommendations. Writing a review of Egypt's history could lead to similar information collected than creating an illustrated book. Though the later should result in other verticals being preferred in the finally recommended item list. As next step the most appropriate sources that fit the underlying task and information need should be selected and ranked. This process that is typically represented in federated or aggregated search, discussed in section 2.2.1, has to be adopted to include the additional information from the user's context. In a federated environment it might even be the case that sources represent only one certain type of vertical at a time. This could also be taken into consideration in the selection process. For example when it can be assumed that the user is primary interested in images, source not holding images can be given lower weights or even be excluded from the results at all. The preselection is crucial specially for systems with vast amounts of connected sources. Without this step each request would have to be forwarded to each single source in the selection. This would not only increase the amount of traffic and load on the service but might even lead to less relevant results in the finally rendered list of recommended items. This leads also to the conclusion that it might not be feasible, as some of the source selection approaches suggest, to forward the request to all sources for gathering the statistical information for the selection process at query time. Therefore every time a source is connected to the federated recommender engine statistical information about its content has to be provided or gathered.

After the list of potential sources is selected for each of these sources a request has to be formulated to retrieve the data. Within this formulation process all the information that is send by the context detection component has to be adopted to meat the sources specific API. Finally the request can be forwarded to the target sources. To be comparable to each other in the next step the returned result lists has to be transformed into a generalized format. Afterwards the transformed result lists have to be aggregated. Here again the aggregation approaches in federated search, discussed in section 2.2.3 can be used or adopted for this purpose. Further several techniques of recommender engines like collaborative filtering, discussed in section 2.1.1, content-based recommendation algorithms, discussed in section 2.1.2 or knowledge based filtering, discussed in section 2.1.3 can be used in this step. In the final step the composed result list can be returned to the user. In the following the single components are discussed in detail. The design of such a system can be divided manly by their tasks into several components that allows the system to be highly modular.

3.1 Context Detection

To gather information of the user that could be used to personalize the retrieval process two main sources of information can be used. I) The history of the user, through logs from the system, or a potential front-end, like the browser history of the user. II) The current context of the user. For example the system can estimate the users intent automatically by analyzing the information the user is currently reading or actions that are taken by the user. All that gathered information can be composed in a profile of the user which can then be send to the federated recommendation engine. The whole context detection component can be designed as completely autonomous module that can be executed directly on the users machine. This approach has several potential benefits. Not only will it reduce the load on the back-end system, it allows also to process more information gathered through the human computer interaction which can finally send to the recommender engine in a condensed form. The possibility to tailor the context detection module towards the environment promotes the adaption to different goals and use-cases. For example in a static environment like a

work place the information of the current location of the user might not be as valuable as in an mobile setting. A further benefit of this approach is that it allows the user to be in charge of the information he is willing to share with the system to improve and personalize recommendations. Even further this option enables the user to circumvent a filter bubble effect [51].

3.2 Source Selection and Probing

As described in section 2.2 in uncooperative settings a preliminary probing phase is needed before adequate sources can be selected. The probing process itself does not necessarily have to differ from probing in federated search. There might be a need of additional extracted or derived information (e.g. meta data or statistical data) from the sampled data that would typically be required. Hence, for example, query based sampling, described in section 2.2.2, can be used as an approach to gather the needed data. The general approach in query based sampling is to send each query of a predefined set of queries to the source. A certain amount of the resulting documents are then analysed to estimate the entire content. Since the system design of the federated recommender engine allows the sources to register it self automatically the probing has to start immediately after the source connected for the first time. The result and details of this initial probing can then be stored for the source selection process. After a certain amount of time, this process has to be repeated to reflect potential changes within the sources dataset. The query set which was used to probe the collections the first time can be reused and extended by keywords extracted from the gathered data. By doing so, the query set should also cover potential trends. Alternatively, the query set could be compiled of highly ambiguous terms to create a similar effect. For select of the set of sources, several approaches can be used in this setting. For example, like described in section 2.2.1, approaches based on lexical statistics can be applied in particular in a combination with the actual query terms. In general information extracted from the content of the sources can be used to reduce the amount of used source. For example sources that primarily contain certain media types (e.g. Images) can be filtered out or prioritised. Furthermore, classification based methods, like described in section 2.2.1, can be utilised to render a set of sources out of

the pool. For example, out of the query, even when generated automatically, a general topic can be derived. This topic can be matched against specialized collections or more comprehensive knowledge-bases like encyclopaedias. Another possibility to limit the amount of sources and improve precision is to take the users history into account. Based on the preceded request of a specific user personal preferences can be extracted and utilized for this purpose.
The source selection module is further one of the locations where unsuitable content can be filtered beforehand. For example, depending on the language level of a user, certain latent attributes can be derived. One possibility would be the estimation a level of expertise of the user in context to the topic of the currently crafted document. For instance recommending scientific literature to a sophomores might be overwhelming but would be appropriate for a professor. Furthermore, a flag for content unappropriated for minors could be set that allows to filter inappropriate content.

3.3 Query Formulation

A new challenge that is arising in the federated recommender setting is the generation and altering of the query to personalize the results. It is generated, adopted and processed in several stages and finally contributes a major part to the quality of the final result list. In the first step the query has to be generated. This can either just happen out of the users context and habits, like daily routines which the context detection module could learn over time, or out of textual information the user is either creating or looking at. The main challenge to extract the needed information out of the text is to detect which terms or concepts are of the most importance. Though, even when assumed that this process works well and only the most important terms are found, there is a high chance that the underlying documents or these terms cover not only one main concepts but several. For the federated recommender system this might lead to problems. When formulating the final query some collections might react to unrelated query terms unforeseeable. For some collections it could lead to a query drift and unrelated results, others could end up not returning results at all. Consequently the core system should be capable of detecting and separating various concepts.

In the case when the system detects multiple concepts it could sent each separated to the collections and merge the results in a later stage.
Another arising problem is owed to the recommendation setting. Depending of the level of knowledge of the user in most cases, for example when crafting a document, returning well-known and obvious information is most probably not the best strategy and will not be helpful to the user. Often it could be the case that topically more diverse results might be more helpful. In other cases the prior knowledge of the user about certain topics could be helpful to in cooperate as well. Therefore it might not be enough to return results with high precision but rather incorporate a certain degree of diversity and serendipity. This can be achieved over several way within such a system of which one is the query itself. For example to achieve such and effect, a diversification of the results, it has to be somehow assured that the most prominent potential intends are covered. Typical example for queries with multiple intends are 'Jaguar', which could either mean the car manufacturer brand or the wildcat, or 'Java' with which the programming language, the island or the coffee could be meant. So, to diversify the results, the system should be able to alter the query in a way which in the end allows to cover the main intends.
Serendipity is usually defined as a discovery that can be described as 'pleasant surprise'. To generate such an effect artificially the system could again alter the query to take the users prior knowledge into account. Concurrently, to create an serendipity effect the system could incorporate the users history which it could get provided by the context detection agent. To do so there are several possibilities, how the users history could be covered. First of all the user should have the option to state his interests, potentially including a level of expertise, directly. Obviously this would be the easiest solution but it may turn out that user potentially would not do so since the connection between improving the results and stating her own interests could be hard to explain. Therefore, the system should also be able to extract similar information just by using the users history, for example the visited pages in a web-browser. In the final step the system should somehow be able to reflect this information within the query. Since it's most probably not possible to generate one single query that is producing precise, diversified and serendipitous results at once the system has to send these queries separately to the collections. The results for each collection and each query type would have to be aggregate afterwards.

3.4 Result Aggregation

As discussed in 2.2.3 typically on of the main challenges in federated search is the result aggregation process. Since the items are retrieved from different collections the individual result-lists have to be unified and presented to the user. This can either happen by showing only a preview of each of this result sets and let the user explore interesting sets further on his own. Another approach is blocking or interleaving where the results are added either in blocks or item by item. A third option is the aggregated search result page where in a first step most relevant verticals are selected and then in the second stage the sequence of the verticals is determined. In federated recommendation engines the aggregation process will be even more complex. First, the aggregation process offers most opportunities to personalize the final result-set for the user. Second, not only results from different collections and verticals have to be aggregated but also results of the same source when several differently formulated queries are send to the collection. This aggregation processes do not necessarily need to happen in the same module. In a first step in the main aggregation process the retrieved items from the collections could be aggregated to one single result list. This could for example happen in a blocked or in interleaved manner as described in chapter 2.2.3. The possibilities of personalizing the result-set by the query is limited to a certain degree. Nevertheless, the context of the user and previously gathered preferences can be used to re-rank the already retrieved results. For example favoured vertical types, like images, could be assigned a higher weight. This information could be extracted as implicit feedback from the users interaction with the system in the context detection module. Further this step allows to filter content that is inappropriate in some way, e.g. not suitable for minors, if the information is available or can be extracted from the items. Before the final aggregation step can be applied a further aggregation has to take place. Since the system might send multiple queries to the collections the returned result lists for these query can also be aggregated. This process of generating the modified queries and aggregating the according results can be done in the specific collection module. This has the advantage that in a federated setting where the collection module could be operated on premise the retrieval time can be reduced and the computational load could be distributed.

4 Experiments

4.1 KNOW At The Social Book Search Lab 2016 Mining Track

In automatic query generation and topic extraction typically several subprocesses are involved. For example, in an initial step, the system can try to identify the potential need for assistance of the user. One scientific challenge that covers this topic is the Social Book Search Lab, which was a track at the CLEF 2016 conference. The goal of the first task of this lab was to identify book recommendation requests in postings on Reddit[1] and LibraryThing[2]. Here users requested assistance from other users to find similar books as described in the posting or the title of a book which they could not recall anymore. The second task within this lab was to identify and link references to books within the text and assign the according ISBN (International Standard Book Number).
The organisers generated three datasets for this task. One containing a total of 4,000 postings from the LibraryThing book requests where about 10 percent were labelled as positive examples. The second one, a total of 250 threads, was extracted from Reddit, more specific from the Subreddits "suggestmeabook", as positive examples, and "books", as negative examples.
The third provided dataset, for the second task, was again taken from LibraryThing containing about 200 initial entries with fifty replies each. The goal was not directly to identify the exact title but assign the correct ISBN numbers of the books to the threads.

[1] www.reddit.com
[2] www.librarything.com

Additionally, the organisers provided a dataset containing book data collections from Amazon and LibraryThing with a total of 2.7 million books. As initial step to gain quick access to the data, it was parsed and indexed into the open source search engine Apache Solr[3].

For the first task, the classification task, the only preprocessing step that was applied on the data was stopword removal. In the second step several sets of features were extracted: First, typical feature based on the textual statistics like the frequency of top n-grams within the samples. Second, tags extracted from the provided book dataset were used to produce further features. Finally, based on the assumption that people asking for book recommendations are more literate and might therefore make less spelling errors in texts the normalized count of spelling errors was added as feature. Additionally, this feature penalizes the postings that contain no decent text. This set of features was used to train three different classification algorithms. A Decision Tree, a Random Forest and a Naive Bayes classifier. A fourth approach, based on the idea of a Veto-Veto [**?**] classifier, was as well reported in the final results. Here, a majority voting schema based on the three trained models was implemented. The parameters of the single classification algorithms were manually optimized.

To approach the second task of this lab, the book linking task, a different initial approach was chosen. Within the dataset one could observed that the titles of the books usually were stated quite precise, sometimes even with the according author's name. This is probably based on the nature of the dataset produced by users that value books. Based on this observation an simple algorithm was implemented. In an initial step, the provided 500 threads and related replies were preprocessed. (e.g. stop-word removal, normalization and removal of special chars) From the indexed data the title and parts of the meta-data like the authors and creators were extracted. This data was preprocessed in the same way as the postings. Finally, the generated data produced a lookup table which also linked to the according ISBN numbers that had to be reported. A simple string matching algorithm was used to find the book titles within the text. One of the major drawbacks of this approach is the high number of false-positive results. Mostly this is

[3] `http://lucene.apache.org/solr/`

Table 4.1: Results of the first run with only a small validation dataset created out of the training data. The results represent the accuracy on the combined datasets of "LibraryThing" and "Reddit".

	Naive Bayes	Decision Tree	Random Forest	Vote-Veto
Accuracy	84.10	78.12	84.09	83.21

Table 4.2: Official results on the testing data. The accuracy on the "Reddit" and "LibraryThing" data are reported separately.

	Naive Bayes	Decision Tree	Random Forest	Vote-Veto
LibraryThing	91.59	83.38	74.82	90.63
Reddit	82.02	76.40	74.16	76.40

based on the fact that some book titles are short, especially when stop words are removed, and contain very general terms. Several strategies to resolve this issue were evaluated although not are applied on the final results. Still, they are reported for the purpose of completeness. The first simple improvement was to remove extremely short book titles, especially titles that contained very general terms which is often the case for books that can be considered to be in the field of life advice. Another possibility is to rank candidates based on certain indicators and put a weight on each candidate. Candidates not reaching a certain weight can be removed. For example, one thing that can increase the weight of a candidate is the co-occurrence of the author's name next to the title. Users often mention the author either before, like "Stephen King's The Dark Tower" , or after the books titles, like "The Dark Tower by Stephen King". Furthermore, a classifier was trained to identify sentences that potentially contain a book title. Here, a supervised approach was used based on a set of labelled sentences in the dataset. Upon this labelled sentences the same feature extraction pipeline used in the classification tasks was applied. Both, authors name based and sentence classification, this approaches may appear to be valid but the results do not reflect them for two reasons. First, the absence of the author's name is a too weak indicator to remove a candidate from the list. Second, the classification accuracy for sentences mentioning book titles was too low to come to conclusions (between 60-65 percent accuracy).

The first figures in table 4.1 shows the results of the first task produced by

Table 4.3: Official results on the baseline versus our approach. The baseline provided used 4-grams as features classified by Linear Support Vector classifier and a Naive Bayes classifier.

	Naive Bayes KNOW	**Naive Bayes Baseline 4-gram**	**Linear SVC Baseline 4-gram**
LibraryThing	91.59	87.59	94.17
Reddit	82.02	76.40	78.65

Table 4.4: Official results on the testing data for the linking task.

	Accuracy	**R**ecall	**P**recision	**F**-score
Linking Task KNOW	41.14	41.14	28.26	33.50
Linking Task LSIS	26.99	26.99	38.23	31.64

the system on a small test set created out of the training dataset from Reddit and LibraryThing combined. Here the Naive Base and the Random Forrest approach perform best on similar level.
In table 4.2 the results of the official test run are presented. The two test sets, LibraryThing and Reddit, were evaluated separately in the official run. For both datasets the Naive Base classifier rendered the best results. In general, the system ranked on place three directly below two baseline approaches provided by the organizers [4]. Table 4.3 shows the official results of the Naive Base classification approach compared to the baseline.

The final Table 4.4 shows the results of the linking task. Here the competing systems were judged upon the F-score. With a score of 33.50 it ranked in the first place.

In context driven federated recommender systems one main component is the context detection model. One main function of such a component is to detect when the user is in need of assistance. The presented experiment shows that, at least in narrowed fields, it is possible to identify a request for assistance in written text with very high precision. Still, in future work, it has to be proven that this kind of approach is generalizable to cover also different and multiple fields.

[4] `http://social-book-search.humanities.uva.nl/#/mining16`

A more comprehensive description of this experiment can be found in [85].

4.2 Query Splitting for Context-Driven Federated Recommendations

In general, the performance of document-based retrieval systems is mainly determined by the quality of the query and its overlap with the actual information need of the user. Though, systems that generate such requests automatically as described in section 3.1 face the challenge to identify the main topic of the underlying work. For example from textual content the user is processing at the moment. Although the algorithms might improve over time it can not be expected that this main topic extraction works always flawless. In such cases, multiple intents might end up in one single query. Furthermore, literature suggests that in certain situations covering multiple potential intents is beneficial if when an exact determination is not possible [58]. Unfortunately not all retrieval systems might respond well to such multi-topic queries especially if they are unusually long. Potentially the retrieval performance suffers in general or the system might not return results at all. This is particularly problematic in a federated retrieval setting where the overall performance relies on the performance of the single collections.
Query splitting in the field of Natural Language Processing is not extensively researched yet. The algorithms rely usually on the usage of query logs or on retrieving initial documents to perform a topical separation. One example of the later is proposed by Yu et al. [81]. In their work they propose three different approaches: 'Relevance-Feedback-Based Clustering', 'Term-Based Clustering' and 'Document- Based Clustering'. 'Relevance-Feedback-Based Clustering' uses a method to group upfront retrieved documents to the initial query terms. The top unique terms of this documents are then again used to create sets of sub-queries. The 'Term-Based Clustering' makes use of Rocchio's query expansion algorithm to assign the documents to each query. Through the cosine correlation the terms are clustered in an agglomerative hierarchical cluster. Finally, the resulting tree is used to build the query term groups. Alternative approaches that rely on query-logs and the identification of latent topics can be found in the work of Ye et al. [80] and He at al. [36].
The in this work presented approach is making use of two recently developed algorithms to generate word embeddings; namely Word2Vec [49]

and GloVe [53]. Both approaches are creating word embeddings whereas Word2Vec is using a predictive model and GloVe is based on a count base model. The approaches as well as the baseline in this experiment do not address the estimation of the amount of topics but rely on prior knowledge. For estimating the amount of subtopics in a query one can refer to the work of Tibshirani et al. [75] or Pham et al. [54].
To create a suitable dataset for this experiment the already well studied Webis-QSeC-10 [34] training set was used. This training set contains 5000 queries extracted from query-logs. Out of this dataset, a new set of mixed topic queries was created where N queries were joined. Here N ranged from 2 to 4.
The first evaluation setup covers the case where for example a system might accidentally identifies two paragraphs containing unrelated topics as one and therefore construct a query with this two topics but in the correct sequence. The second evaluation setup covers the case where the topically related order is not given anymore. This could be the case when extracted terms were first weighted by there importance before they were sent to the system.

Baseline: To gain insights of the complexity of the task and the performance of the word embedding approaches a simple baseline was defined that should render near-optimal results. This approach simply splits the composed query string into k equally sized chunks. Since most queries have the tendency to be between three to five terms [2] this approach should render good results in the first evaluation setup. However, in the second evaluation setup, the results are expected to be of low quality.

Word Embedding Based Classification: The algorithms either used the vectors of a pre-trained Word2Vec model, from the Google News dataset, or from a pre-trained GloVe model, a Wikipedia dump of 2014 combined with the English Gigaword Fifth Edition newswire text corpora. This vectors where then used to group the query terms with the clustering algorithm K-means. Here k, the number of clusters, was set to the number of merged queries.

Two measures were chosen to report the clustering performance: V-measure [60]

Table 4.5: Rand Index results of the two splitting approaches within the first scenario where the position is taken into account. The values descent from iteration to iteration for each of the approaches.

	Two Queries	Three Queries	Four Queries
Word2Vec Kmeans Rand Index	0.710	0.643	0.595
GloVe Kmeans Rand Index	**0.729**	**0.697**	**0.648**
Split Approach Rand Index	0.717	0.664	0.631

Table 4.6: V-Measure results of the two splitting approaches within the first scenario where the position is taken into account. The V-Measure values seem to be almost independent of the number of mixed queries.

	Two Queries	Three Queries	Four Queries
Word2Vec Kmeans V-Measure	0.770	0.770	0.769
GloVe Kmeans V-Measure	**0.788**	**0.806**	0.780
Split Approach V-Measure	0.775	0.781	**0.789**

and Rand Index [38]. V-measure represents the harmonic mean between homogeneity and completeness of two clusters. The returned figures are between 0 and 1; higher values are better. The Rand Index returns the similarity of two clusters through all the pairs of samples. It returns values from -1 to 1 where 0 represents randomness, 1 full correlation and -1 negative correlation.

Table 4.5 and table 4.6 show the results in the first scenario where the sequence of the query tokens are in the correct topical order. In general, the GloVe K-means approach rendered the best results, although figures of all approaches are nearly on a similar level.

Table 4.7 and table 4.8 show the Rand Index and V-Measure results for the second scenario where the position is not taken into account. As expected the result of the splitting approach, the baseline, are low in general and

Table 4.7: Results of the two splitting approaches within the second scenario where the position is not taken into account. The Rand index seems to decline with each iteration. The highest values are achieved by the GloVe based approach. The Rand Index results of the splitting approach indicate a random assignment to the clusters as expected.

	Two Queries	Three Queries	Four Queries
Word2Vec Kmeans Rand Index	0.088	0.071	0.056
GloVe Kmeans Rand Index	**0.281**	**0.232**	**0.199**
Split Approach Rand Index	0.008	0.003	0.000

Table 4.8: Results of the two splitting approaches within the second scenario where the position is not taken into account. Here highest values are produced by the Glove based approach, which even raises by each iteration. For the Word2Vec based approach, the values seem to be almost stable, independent from the number of mixed queries. The lowest results are achieved by the splitting approach.

	Two Queries	Three Queries	Four Queries
Word2Vec Kmeans V-Measure	0.373	0.341	0.373
GloVe Kmeans V-Measure	**0.427**	**0.477**	**0.502**
Split Approach V-Measure	0.278	0.267	0.236

the Rand Index results indicate random behaviour. The Word2Vec based approach already creates better results, but again the GloVe based algorithm renders best results for both measures by far.

The results of the first scenario indicate that there is one major factor. As expected the average query length seems to the dominant factor which explains why the basic splitting approach renders decent results in this scenario. Since the query length is mostly similar on average splitting might just be off by one in many cases. Even in that scenario the GloVe based approach outperforms the others slightly.
However, in the second scenario the splitting approach, as expected as well, generated random assignments to the term groups. Here the Word2Vec approach generates better results. Though, the GloVe based approach produces the most reasonable results.
In general, it could be expected that using a dedicated Word2Vec or Glove model could improve performance further when trained on query log datasets. Furthermore, other classification algorithms, for example, deep learning based approaches, could be beneficial as well which could be evaluated in future work.

Preprocessing an automatically generated query is an important task in a service based federated recommendation framework. Not only is it necessary to understand the user's intend, it's also important to identify potential errors introduced by the context extraction process like the generation of multi-topic queries. For example, through this process the results returned to the user can be improved by covering several different topics on one single result list. Furthermore, it can prevent that system that cannot cope well with multi-topic queries do not reply at all. Which could lead to valuable information not reaching the user at all.

A more comprehensive description of this experiment can be found in [83].

4.3 Evaluation of Pseudo Relevance Feedback Techniques for Cross Vertical Aggregated Search

One of the underlying challenges of cross-vertical aggregated search and therefore likewise for query-based federated recommendation systems is the presentation of aggregated result lists to the user. Though, evaluations in this setting are complex. Evaluations in information retrieval often follow the Cranfield paradigm [26]. It relies on a crafted data set of items or documents where the relevance or importance of the documents according to a query is judged by humans. Thus, one can judge the performance of a retrieval system through measures like mean average precision(MAP), normalized discounted cumulative gain (NDCG) and others in an offline manner. Although this approach has been proven to be valuable, there are certain limitations to it. An extensive discussion of this topic can be found in the work of Voorhees [77]. One of this limitations is that for certain aspects that are valuable in some situations this approach does not seem to be applicable. An example would be settings where aspects like diversity and serendipity play a role which could be considered of higher importance in recommender like system. Particularly for serendipity, it would be hard to gather ground truth data and measure the performance of an algorithm since this effect depends on the personal experience of the single individual. The first goal of the in this work presented experiment was to create an experimental setup that allows evaluating algorithms in a federated search setting where the Cranfield paradigm cannot be applied easily. The second goal was to get information about type and amount of sources used in federated retrieval when pseudo-relevance feedback should be applied. Pseudo-relevance feedback is a technique to improve the relevance of the retrieved documents by appending terms to the original queries. To do that, the initial query is send a first time to the collection. Out of the top search hits of the result set of this initial search query term, candidates are selected. This query term candidates are finally appended to the initial query and, after a second retrieval step, the final result set is returned to the user. This technique is of interest in our setting since the query the system receives will be auto-generated. Hence, there is a possibility that the

query does not cover the information need of the user optimally. According to literature one option to resolve such a problem is result diversification with the means of query reformulation [63]. Hence, for the reformulation process, pseudo-relevance feedback algorithms were used to expand the query and create a query drift towards most common topics. Query drift denotes the effect when a query is altered to an extent that it causes a semantic drift within the result set [42]. Other parameters to take into account are the amount of retrieved items to consider for the key term extraction process and the number of query terms used to add to the query. For the first parameter Montgomery et al. [50] suggest using the top ten documents. The second parameter, the amount of added query terms, should be limited to a maximum as well since precision drops after exceeding a certain amount [35]. For this experiment, an upper limit of twenty unique query terms was chosen. In a federated setting Shokouhi et al. [68] demonstrated that only using a set of predefined sources instead of all available sources could be beneficial. On the other hand, Lynam et al.[46] stated that the suitability of a source for other sources could be estimated by the performance impact when used on itself. This might also imply that a single source for the expansion process, for example, a comprehensive knowledge-base, could be a viable option. Finally, this led to the three different strategies that competed against the basic unaltered result list.

No Query Expansion As baseline, the query was sent unexpanded to the sources.

Multiple Sources This setting takes all sources into account. Here the query was sent first to all the sources. The top documents of all the sources were the basis to for the selection of the expansion terms. Finally, the expanded query is sent to all sources.

Single Source Although similar to the multiple sources approach, this takes only one single preselected source into consideration. The source was selected based on its performance when queried with its own expanded terms like described previously.

External Knowledge Base The last setting is making use of a knowledge base as expansion source, namely Wikipedia. The knowledge base

itself is not part of the finally queried sources.

To generate the expanded query a simple strategy was used. The original query terms were conjunct while the expansion terms were added in a disjunctive manner.
Although there are several different options [41, 6] to aggregate results from different sources a simple interleaving algorithm was chosen to not create a potential bias. Therefore the results were combined by picking the top result of each list in a fixed sequence ascending.
For this initial experiment, a dedicated evaluation tool was created to have full control of all parameters. One of the lesser goals was also to gain insights how to design such an evaluation to be conducted on crowd-sourcing platforms. This tool presented a fixed query to the user, optionally a short description of the query, and the four result sets according to the four strategies. The order of the results lists were randomized to avoid a bias. The user could grade each of this result sets from one, the best result, to four, the least fitting result, by clicking onto the result list.

The actual task of the user was to compare the presented search results and rank them accordingly. Once the sequence was selected, the user got a new query with associated result sets. The used query and result dataset were composed beforehand since it had to be ensured that the system works in a deterministic manner. All the users evaluated exactly the same queries with the same result lists. The basis for the queries was the AOL query log [52] from which top queries were extracted and further individually selected. Each of the final 20 test subjects was instructed personally, and notes about comments and feedback were taken during the evaluation. Interested individuals were given a short background of the system, but no hints were given by which means the search results should be rated. Figure 4.1 shows the dedicated evaluation tool.

The result of this evaluation can be utilized in several ways. First, from the recordings of the tool a qualitative and quantitative evaluation of the approach can be made. Second, the notes taken during the session can be

Figure 4.1: The figure shows the dedicated tool for the evaluation. The tool presented the query, a description of the query and the four result set randomized to avoid bias.

Expansion Method	Overall Score	Entity-Centric Queries	Topical Queries
No QE	431	**103**	328
Multiple Sources	466	129	337
Selected Source	427	143	284
Wikipedia	**426**	155	**271**

Table 4.9: Results for all four query expansion strategies, where the number indicates the accumulated rank, thus lower values are better. Results are also separated into entity-centric queries and topical queries, where the first type of query refers to individuals, organisation, and other types of entities.

used to render a guideline for evaluation setups in the future.

Users reported that it was often hard to render a decision based on the given result lists since results were often quite similar. This is also reflected by the Krippendorff's Alpha measure of disagreement which lies between 0.66-0.78. This value can only be considered as 'fair' agreement. This outcome was probably affected by several factors. Some participant only considered the top results while others measured the overall result-set performance. This is also reflected by the evaluation times per query of about one to five minutes. Furthermore, since the queries were always presented in the same order and not randomized a fatigue towards the end of the evaluation can be assumed.
Table 4.9 shows the quantitative results of the experiment. The given figures represent the accumulated rank of the user's decision, therefore lower values indicate a preference towards the given approach. The expansion methods based on Wikipedia and the preselected sources yield slightly better results in general than the baseline with no expansion at all. Though, when inspecting the query types more closely, there seems to be a tendency that topical queries seem to benefit most from the query expansion. Entity-centric queries, queries where one would expect one single Wikipedia page like "Michelle Obama", seem to perform better with no expansion at all. In conclusion, the expansion process using one single knowledge base, in this case, Wikipedia, seems to be the most beneficial. However, in certain cases like entity-centric queries, it should be considered to have no expansion or have a mixed approach with unexpanded top results and expanded results aggregated in the lower ranks.
Finally, the observed behaviour of the participants allowed to derive a set of

criteria to design such a user study on a large setting like on crowd-sourcing platforms:

- The rating strategy how to evaluate the results should be defined and explained clearly. (e.g. full set versus top results)
- A maximum of two sets of lists should be compared against each other to facilitate rendering a decision.
- When diversity and serendipity in results set should be evaluated the full result set has to be compared.
- The amount of queries judged per user should be selected carefully to avoid fatigue.
- A further measures to reduce the consumed time and needed attention could be to present only the title of the items and no snipped.
- A short questionnaire at the end of each task could improve the overall quality and allows to gain further insights into the decision rendering process.

This experiment gives first insights how to render result lists in the setting of federated recommender engines in an efficient manner. Furthermore, it provides insights for larger user based studies like described in section 4.6.

A more comprehensive description of this experiment can be found in [82].

4.4 Do Ambiguous Words Improve Probing for Federated Search

One of the most crucial steps in the whole federated search process is the source selection step. The task of finding matching sources towards a user request, a query. To facilitate this, typically detailed information is a necessity. For example information such as meta-data and statistical information of the collections corpus. Federated search in an uncooperative setting, were detailed insights into the corpus are not available and documents can only be retrieved by a potentially restrictive API, faces the challenge of collection representation. This is the case especially when the system is designed in a freely accessible way, and collections can be added and removed dynamically at runtime. In recent years, to face this challenge, the most common approach, in this case, is called "query-based sampling". Further details can be found in chapter 2.2.2. In query-based sampling the system sends requests to the collections and gathers the needed information from the results returned. In such a dynamic system the goal would be to make the collection available as fast as possible after the first connection to the system. Though, an ideal solution should send only a very small set of queries to the collection to get a realistic image of the underlying information.
Although, several proposals were made, as discussed in section 2.2.2, how the query set should be created a method relying on ambiguous words hasn't been explored yet.

The here presented work tried to assess possibilities for a new category based source selection approach while reducing the amount of needed probing attempts to a minimum. The collection selection approach used for this evaluation setup is based on a category mapping algorithm. This algorithm takes a set of terms (t), a query (q) or a document (d^s) from a source (s) of all sources (S), and maps it to a set of categories (C).

$$f(\{t\}) = \{< c, w_c > | c \in C\} \tag{4.1}$$

To each category c there is a weight w_c applied to rank the candidates afterwards.

Input Terms	Output Categories
battle of trafalgar	{ military, history }
women wage gap	{ economy }
world cup football	{ sport }
java 8 features	{ computer_science, food }
dinosaur t-rex	{ animals }
sentimental tears emotions	{ psychological_features }
kittens for sale	{ animals, commerce }
department of justice	{ administration }

Table 4.10: Queries given to the algorithm to assign domains. In general the results seem to meet the expectations.

Through this ranking, it is finally possible to compare different mappings and render multiple sets of documents.

As source of the used categories was extracted from WordNet Domains [5] [47, 9] WordNet contains most English terms organized in sunsets which are normally assigned to domains in WordNet Domains. This Domains model a tree-like hierarchy containing a total of 164 nodes. Additionally, WordNet Domains has the concept of factotums which represent synsets with a high number of categories. The algorithm is making use of this dataset by extracting these categories for each given term. Further, in the initial step, each term gets the same weight assigned. In a second step, the weight is propagated to its parent notes through a decay function of $1/n$ with n giving through the distance from the leaf note to the parent note. Through aggregating the weights over the input a set of domains for this input is created. The top 5 domains are finally chosen to represent the categories of the input. Example results can be seen in table 4.10.

One dataset that is dedicated to Federated Search research is the "FedWeb greatest hits" dataset [27] consisting of about 800 GB crawled HTML pages from about 200 different collections. Through an analysis of the FedWeb dataset, it became apparent that in general sources could be categorized

[5] http://wndomains.fbk.eu/

into two major groups. Encyclopedic sources, which cover a wide variety of content, and niche sources, covering specific thematic content. For example, containing exclusively documents about biology.
Since one hypothesis was that choosing ambiguous terms for the probing process to improve the process four different query selection strategies were developed.

Factotum Queries
: These queries were produced through extracting so-called factotums, multi-domain word, from the WordNet synsets.

Ambiguous Queries
: Similar to the factotum queries these queries were constructed from multidomain words but with the minimal limit of three domains.

Random Queries
: Queries constructed by randomly drawing from WordNet.

Random Queries without retainment
: Here the words were not retained between consecutive probing attempts.

To conduct the evaluation three main questions were addressed. First, the question if the algorithm detects all domains. Second, is the algorithm producing stable domain sets results? Third, is it really applicable to sources where the domains are not known beforehand.
To address the first question a small dataset consisting of 3 blocks with 20 documents where each block was coming one from a specific domain. These selected domains were mathematics, religion and health. Finally, the dataset was passed to the algorithm. The results are presented in Figure 4.2.

The algorithm identifies the domains mathematics and religion correctly. The domain health is missing, although the third top domain in the results is medicine. In this case, it might be due to the WordNet Domains. In general, as long as this mapping is consistent it should not interfere with the results of the source selection approach.

To address the second question, if the domain set produced by the algorithms stabilize over time, the FedWeb Dataset was utilized. The most

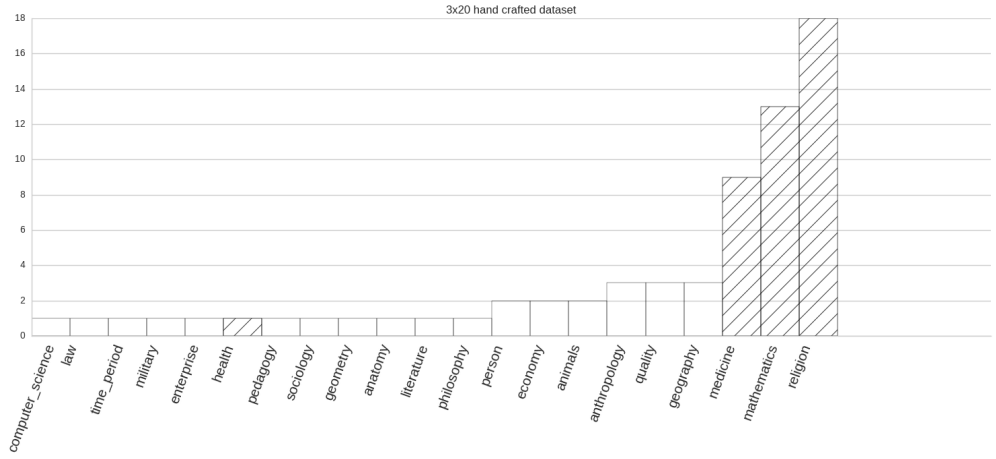

Figure 4.2: The plot shows the results that were detected on the constructed dataset. Except for the domain medicine which replaced the domain health the assignment of the domains seem to work as expected.

fitting measure for this task seemed to be the "rank biased overlap" [79]. This measure employs the hyper-parameter p, reflecting the importance of the top results, which was set to 0.8. The algorithm was applied to this dataset following our four query drawing strategies. Drawing from 100 to 2,000 queries in steps of 100s. Figure 4.3 shows the results of this approach.

Like described before, two groups of the sources could be seen in the dataset, niche and encyclopedic sources. Since it was of interest to which extend the observed behaviour of the algorithm changes in comparison to a random baseline the dataset was split into these two groups. Figure 4.4 shows the results of this evaluation with the mean and standard deviation between the curves of this two groups.

To address the third question, the final verification of the algorithm, the last experiment was conducted. For this experiment, another source was taken, in particular, ZBW [6], a collection of scientific economic literature. As comparison dataset from 100 queries, the top ten documents were manually assigned to sets of domains. Results can be seen in Figure **??**. All algorithms agree on the top domain, economy. In the rest of the domain sets, overlaps can be spotted but not with the same rank.

[6] `http://www.zbw.eu/`

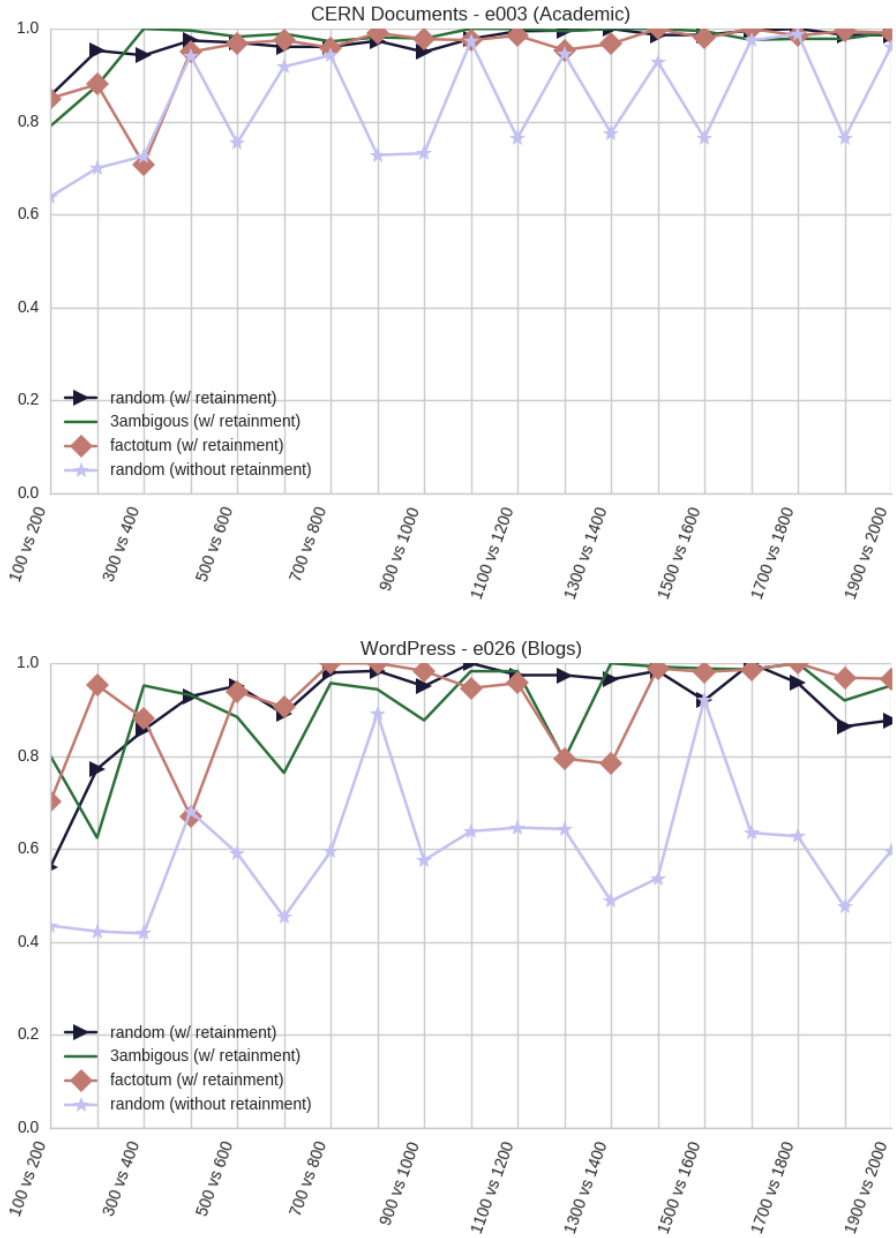

Figure 4.3: The upper plot an example graph on a niche source called ”CERN Documents”. The upper graph shows the results from a niche source ”CERN Documents”. The second one is the result of an encyclopedic source,”WordPress”, in comparison. In general, the approaches yielded stable results faster on niche sources. As expected the lowest and most unstable results are given by the ”random without retainment” approach.

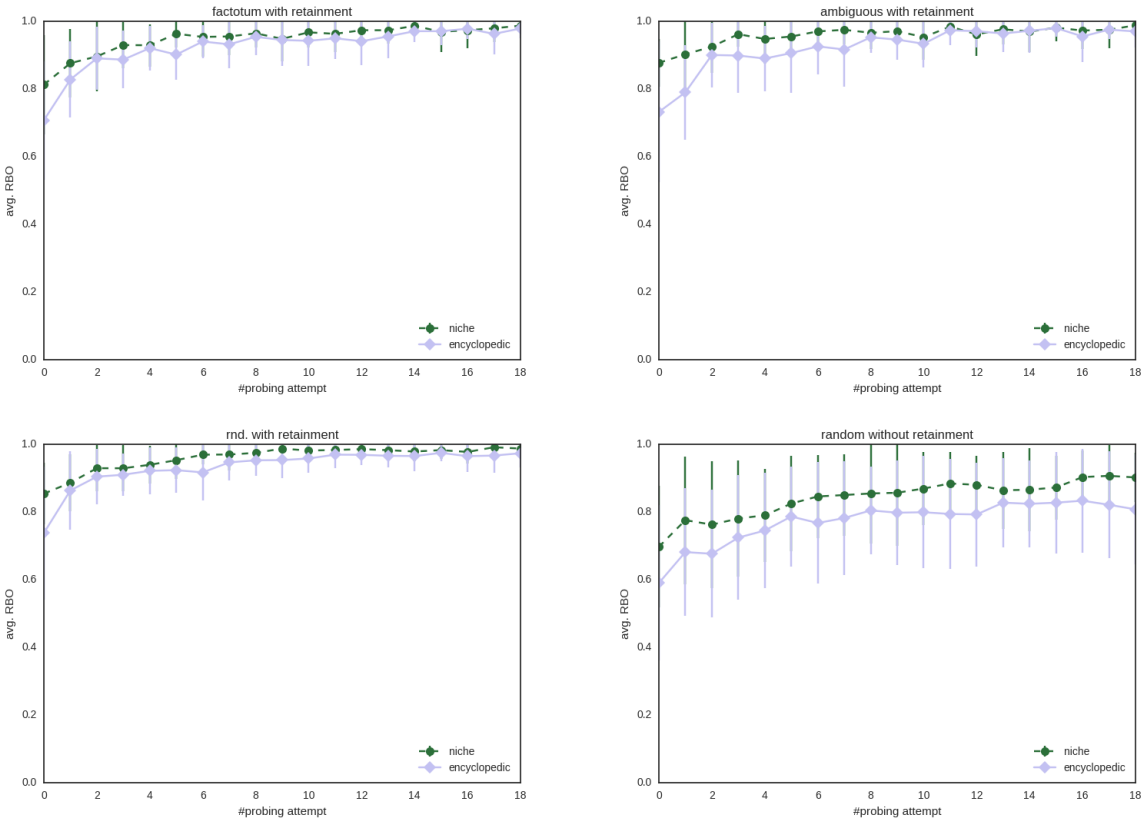

Figure 4.4: The plots show the results of the different query generation methods. Each plot represents the mean performance of the single generation process separated on the niche and encyclopedic sources. In general, the factotum, the ambiguous and random query generation processes show a similar performance, only random without reattainment has a high deviation from the mean.

For a federated recommender engine, one important factor to bring benefit for the user is the amount of connected collections. It can be assumed that a high number of available resources allows the system to tailor the results better towards the user's information need. That means in an ideal setting connecting to the core of the system should be of low effort. This also implies that getting standardised statistics about the joining collection directly from the collection itself is most properly unrealistic. Hence, it can also be assumed that in such a system collections join an drop out frequently. The more often new collections join, and the more systems are registered, the more important is an efficient probing approach. In this context, the results of this experiment are highly relevant for the system. Still, in that regard, more experiments would be needed.

4.5 Efficient Search Result Diversification via Query Expansion Using Knowledge Bases

As described in section 3.3 and section 3.4 one step towards a federated recommendation engine is to incorporate diversified results in the final result set instead of returning solely highly precise results.
Usually, queries sent by users to a search engine tend to be short and ambiguous [20, 55]. A search engine receiving this queries therefore might want to cover every potential interpretation of the query to cover the user's information need. One possibility to cover all these meanings is diversification of the search results. A comprehensive overview is given by Drosou et al. [29]. In their work, they identified three major groups of approaches. Content-based, novelty-based and coverage-based where in all these categories a subset of results is selected with the aim to maximize diversity. Usually, this restructuring is performed as a greedy approximation.
One downside of this algorithms is that the retrieved result list has to be significantly larger than the amount of returned result to cover all potential interpretations. This is especially a drawback when the search engine is located remotely due to higher response times for longer document lists. Further, processing and re-ranking these items can be computationally expensive. Therefore, it would be beneficial if this process could be simplified or avoided. But in the case when the search engine is not directly modifiable this steps can not be avoided when diversification is achieved through result list re-ranking. The only feasible alternative is to alter the query before it is sent to the system. One example of such an alteration of the query would be query expansion. Here the query is extended with terms that are related or are synonyms of the original query terms. In their work Clark et al. [25] described that diversity increased through pseudo-relevance feedback which is a related technique. Simplified and in brief, a query is sent to a search engine. The top results are retrieved and analyzed, and the top terms in this results are added to the query. Therefore, query expansion seems to be a feasible option. Still, it is an open question to which level diversified results can be achieved through this process. Furthermore, how to surrogate the retrieval process of the top documents which again would add retrieval time and computational effort.
Latency and computational effort can only be reduced in a federated search

system when pseudo-relevance feedback based query expansion is performed on a local search engine instead of an external collection. Therefore, a dedicated index containing content of the English version of the Wikipedia was created. With the help of this search index, a list of expansion terms was crafted by sending the unexpanded query to this search engine and extracting expansion terms out of the top results. Details to the indexing and term extraction process can be found in [61].
Finally, the extracted expansion terms were added to the original query in a disjunct manner. All expansion terms where added as a single query clause to avoid a too distinct query drift:

$$OrigQueryTerms\, OR\, (ExpTerm_1\, OR\, ...\, OR\, ExpTerm_n)$$

The queries generated in this way were then finally sent to the underlying collection.
As dataset for the initial queries a dataset extracted from query logs contributed by Seifert at al. [65] was used. Out of this dataset, 70 queries were hand selected to gain a clean query-set.

As baseline against the query expansion approach the well known IA-Select [1] algorithm was chosen. IA-Select requires every query and every document to be assigned to a list of categories. To full-fill this requirement categories given trough the Wikipedia category graph[7] and DBPedia [8] were used. Each query got manually assigned to appropriate categories while for the documents the already defined categories were used. Since the Wikipedia category graph is extensive a mapping schema towards the main Wikipedia categories was applied.
To assess the level of diversity of the result lists the NDCG-IA (Normalized Discounted Cumulative Gain - Intend Aware) [1] measure was used which is the intend aware version of NDCG.

One major parameter in this evaluation is the amount of added expansion terms to the query. For the first results, presented in Figure 4.5, the amount

[7] http://data.dws.informatik.uni-mannheim.de/dbpedia/2014/en/skos_categories_en.nt.bz2

[8] http://www.dbpedia.org

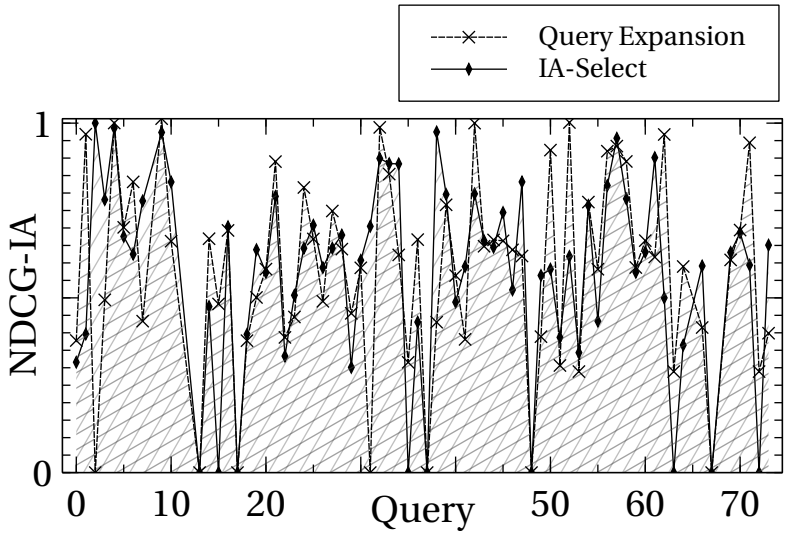

Figure 4.5: Raw results of IA-Select and the query expansion algorithm with their respective NDCG-IA@10 values for each query. Although in most cases the NDCG-IA results appear to be similar, in some queries the two values disagree.

of expansion terms was set to 10. The figure presents the values of the NDCG-IA@10 for both, the IA-Select and the query expansion algorithms. Here, most results are similar, but for some queries, the algorithms disagree.

Figure 4.7 shows the results of the two algorithms as scatter plot for NDCG-IA@10 with ten expansion terms for the query expansion approach. In general, the two distributions seem to be positively correlated.

One of the relevant questions is the optimal amount of needed expansion term to reach a certain degree of diversity. Figure 4.6 shows correlation between IA-Select and the query expansion method from 5 to 20 expansion terms. The provided correlation measure results are Pearson's r, Spearman's rho and Kendall's tau. The highest Pearson's r correlation can be observed at 20 expansion terms. For Spearman's rho and Kendall's tau the highest correlation can be seen at ten expansion terms.

In general, it seems to be the case that query expansion based on an unrelated knowledge base can be used to diversify results on a similar level than

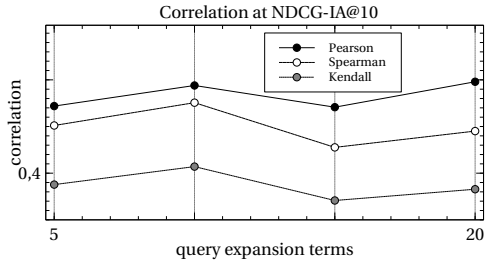

Figure 4.6: Correlation values of the NDCG-IA@10 for the query expansion and IA-Select approach in relation to the count of the query expansion terms. The highest Pearson correlation occurs at term expansion $numExpTerms = 20$ with $r = 0.60$. Altogether the coefficients indicate a better correlation for the query expansion with 10 terms.

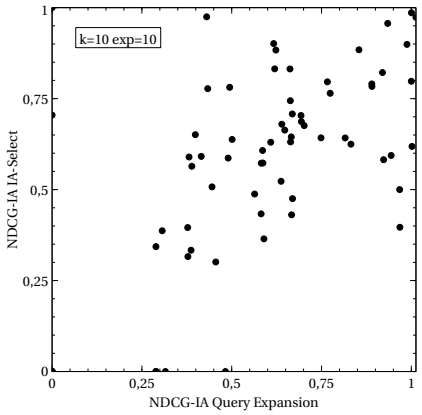

Figure 4.7: Scatter plot of NDCG-IA@10 for the two diversification methods. Each occurrence o is defined as $o = (NDCG\_IA@10(R_{QE}(q_i')),\ NDCG\_IA@10(R_{IA}(q_i)))$ with $q_i' = QExp(q_i, 10)$. The two distributions appear to be positively correlated.

IA-Select. Though, the amount of expanded terms is a key factor for the performance of the approach. According to the results, it can be assumed that the best choice diversity wise and performance wise is at about 10 expansion terms when added to the original query in a disjunct manner.

For a federated recommender system, these insights are relevant on several levels. First, it allows diversifying the results without retrieving more data from the connected collections then necessary which avoids higher response times. Second, since for diversification altering the query is beneficial, it can be assumed that a similar method can also be utilized to create other

effects. One example could be result list personalization which is discussed in section 4.6.

A more comprehensive description of this experiment can be found in [61].

4.6 Evaluation of Contextualization and Diversification Approaches in Aggregated Search

As described in section 2.2.3 there are several approaches to resultlist aggregation. All these approaches can be categorized into three major groups: ”Non Blended”, ”Blended” [73] and ”Composite Retrieval” [12]. The ”Non-Blended” approach show for each collection an own separated view. In contrast, the ”Blended” approach takes all the results from all collections and returns one single result set to the user. While ”Composite Retrieval” is a mixture of ”Non-Blended” and ”Blended” approaches.
In the setting were the system should present diverse and serendipitous together with unaltered results the most sensible approach seems to be the Blended approach. Still, there are several options how to blend different result list into one single set. Two of the most prominent methods, when it comes to highly diverse content in regards to media types or vast amounts of different collections to be aggregated, is arranging the results of the different collections in blocks [5] or as interleaved list [23]. These two are of special relevance when ranking values are not given or are not comparable to each other. Still, according to Arguello et al. [3] there is no user study that compares these two approaches yet.
In a previous study [61], described in section 4.5, an approach for diversification on query level by the use of a pseudo-relevance feedback technique was evaluated. To generated serendipitous results a similar approach was used. From a query log set of the EEXCESS [9] project [65] with the according to users web history user profiles were extracted. By using the according web-pages as a result set for our pseudo relevance keyword generation process, it was possible were able to extract top keywords representing the user’s history. These keywords where added to the query in a conjunctive manner to generated an artificial query drift towards this topics.
An example of these two query formulation approaches can be seen in figure 4.8.

Based on the insights of a previous user study [82] an evaluation on a crowd-

[9] http://eexcess.eu/

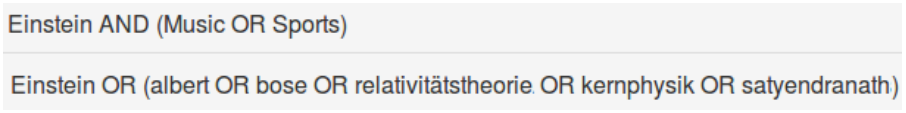

Figure 4.8: In this figure two queries are presented. The top row shows a query aiming for serendipitous results. The expanded terms form a disjunction with each other and are conjunct with the original query terms to produce an intentional query drift. The lower row shows query that aims to produce diversified results. Here the expansion terms are disjunct as well to boots under-represented topics.

sourcing platform CrowdFlower[10] was conducted with over 300 workers participated producing more than 1500 judgments.
The previously described user profiles with the according to queries and the user's history was finally used to render three results sets: the basic unaltered result set, the diversified result set and the serendipitous result set. With the help of this three query datasets a total of five evaluation result sets were created which were finally used in the evaluation:

Basic The baseline; containing only the items returned by the system for the unaltered query.

Interleaved A result list where the top results of the basic, the diversified and serendipitous results were interleaved one by one.

Blocked In the blocked list the first block contained the top results of the basic list, the second block the top results of the diversified list and the third block the results of the serendipitous list. All results were de-duplicated against the already selected results in the final result set.

Diverse Like the blocked list but leaving out serendipitous results as the third block.

Serendipitous Similar like the diverse list but containing only basic and serendipitous results.

These evaluation-sets were the basis to gain insights in the following research questions:

- How does the block ranking perform against the basic result list?
- How does the interleaved list perform against the basic result list?

[10] `https://www.crowdflower.com/`

- How do both approaches perform compared directly against each other?
- Can serendipitous results be achieved by query reformulation on a similar level than diversification?

To answer this questions a total of three evaluation setups on the crowdsourcing platform CrowdFlower were created. In each of this setups, the workers got several pieces of information to perform the task. A comprehensive instruction and how to evaluate the given results, the query, a link to google for background information of the query, information about the history of the user, the two result lists that had to be compared and two question two the workers. The first, asking which list the worker found more suiting for this setting. The second, asking about the confidence level of the user. This setup was the result of a short pretest on the platform was the feedback given by workers was incorporated to improve the evaluation. (e.g. the feedback about the comprehensibility of the introduction) The previously mentioned five evaluation-datasets were used to create a total of three evaluation scenarios:

Scenario 1 This scenario aimed to get a basic understanding of the acceptance of blocked or interleaved results. Here, the workers got either to compare the unaltered result list against the block or the interleaved result list.

Scenario 2 In this scenario the workers had to compare the shortened basic list against a diversity blocked list or a serendipity blocked list. Here, the main goal was to see if one of the two approaches has a severe adverse effect on the other.

Scenario 3 Here, the blocked results were directly compared to the interleaved results. The goal of this scenario was to eliminate the possibility that it proves to be too difficult for the workers to get into the mindset of potential users. The results of this scenario should allow to still draw conclusions about the performance of this approaches.

In such setting typically measures like Fleiss' κ [31] for inter-rater agreement is not applicable since the there only a small set of overlap from users and

	Item Agreement	Decision Percentage
Interleaved	0.692	0.358
Blocked	0.721	0.355

Table 4.11: Results for the interleaving and blocking approach. Reported values are results of the arithmetic mean agreement on item level and the decision percentage towards the stated approach within the corresponding row.

queries. Though, the reported data is the agreement on item level with the arithmetic mean of the percentage of the biggest agreement. See equation 4.2 as illustration. S is the set of tasks, and a_i and b_i are the sums of votes towards one of the algorithms while a_i is bigger than b_i.

$$f(x) = \frac{\sum_{i \in S} \left(\frac{a_i}{a_i + b_i} | a_i \geq b_i \right)}{|S|} \tag{4.2}$$

The goal of scenario #1 was to gain insights into the general acceptance level of the interleaved and blocking approach. Table 4.11 shows the direct comparison between the basic lists and the interleaved or blocked lists. The worker's agreement is at about 70 percent for each of the approaches. While analyzing the queries, it came apparent that only one query was present in both sets. Though, it seems that both approaches are justifiable but do yield different user satisfaction. The decision percentage is on the same level for both approaches at about 35 percent. Since the given query and users history did not reflect the workers, personal information need this results can be considered as sufficient.
In scenario #two the potential adverse effect of the diversity and serendipity approach on each other was assessed. The results in table 4.12 on the item agreement and decision percentage indicate that both algorithms work on similar levels. Further, it can be concluded that the approach to generate serendipitous result list seems to be as feasible as using pseudo-relevance feedback for diversification.
The goal of scenario #3 was to eliminate the possibility that the workers find it to be difficult to get into the user's mindset by comparing the two approaches directly. Table 4.6 show that there is a slight tendency towards the blocking approach. Although, this could also be the result of the nature

	Item Agreement	**Decision Percentage**
Diverse	0.769	0.307
Serendipitous	0.746	0.31

Table 4.12: Results for the shorter blocked lists containing either only diversified and basic results or serendipitous and basic results. Both approaches have similar agreements and decision percentage.

	Item Agreement	**Decision Percentage**
Blocked vs Interleaved	0.647	0.532

Table 4.13: Results for the interleaving approach compared directly against the blocking approach. Both approaches obtain similar result, though a slight tendency towards the block ranking approach seems to exist.

of the blocking approach which takes as first block the top results of the original result lists.
Finally, table 4.14 reports the confidence level of the workers after rendering their decision.

In a Federated Recommender Engine where the goal is not only to return highly precise results but also diversified results while taking the users history into account the aggregation process of the results is a key factor for user satisfaction. The results of this experiment indicate that the decision to use interleaving or blocking approaches might depend on the preferences of the user and on the query that is sent to the system. The approach to

	Interleaved	**Blocked**	**Seren.**	**Diverse**	**Blocked vs Inter.**
Very Conf.	30.0	34.0	47.5	42.5	27.0
Confident	52.0	47.0	38.0	41.5	59.0
Hard	18.0	19.0	14.5	16.0	18.0

Table 4.14: Results of the feedback of the users how confident they were with their decision between the pair of lists in percent. Each user had to state his confidence for each task.

create a serendipity effect by query expansion based on the users browsing history appears to be sensible but more research on this topic is needed.

A more comprehensive description of this experiment can be found in [84].

5 Conclusion

This work aimed to answer the question how to create a system that tries to auto-respond in a personalised way to an information need of the user while relying on a vast amount of diverse sources. To answer this initial general problem several research questions were formulated:

1. Which means are necessary to extract topics out of the textual context?
2. What challenges arise from automatically generated queries and what are potential methods to master those? Additionally, which query processing steps are beneficial in this setting?
3. Can collection representation be further optimised especially for uncooperative settings?
4. How can distributed document retrieval be personalised, in particular when highly precise results are of lesser importance? Furthermore, which aggregation techniques are beneficial in this setting?

According to these open problems, six experiments were conducted resulting in one publication each.

The participation in the social book search lab was aiming to answer the first question. The results of this lab indicate that detecting the situation when users are in need of assistance is possible with high accuracy. Although the lab aims just to detect one specific situation, it is likely that the approach could be used applied in a more general setting.
While working on question two it became apparent that one of the challenges in automatic query generation are the multi-topic queries potentially created by an automated generation process. In a federated recommender system where collections might not respond well to such queries an approach would be needed to disjoin those queries topically. To identify multi-topic queries an approach utilising word embeddings seems to be feasible. Still,

the approach can very likely be further improved by relying on more complex classification algorithms an dedicated word embedding models. This query splitting can be done in an own module in the system. Within this module additional processing steps can be included as well. For example, a duplicate of the original query can be expanded to diversify the results or terms extracted from the browsing history of the user can be added to create a serendipity effect in the final result list.
Collection representation in such an uncooperative setting was evaluated in two ways. First, with the help of WordNet Domains, an efficient approach for collection selection can be implemented. Here, the main challenge is to identify encyclopedic sources early where the assignment of domains varies strongly even after several probing attempts. Second, choosing ambiguous or so-called factotum queries does not necessarily reduce the number of queries needed in the probing process.
Relying on the users browsing history and pseudo-relevance feedback are means that can be utilised to improve and personalise results in such a setting efficiently. According to the conducted crowd-sourcing evaluation in this setting block ranking based aggregation seems to the most beneficial although in some instances interleaving of the result items is a good alternative.

Bibliography

[1] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *Proceedings of the second ACM international conference on web search and data mining*, pages 5–14. ACM, 2009.

[2] Avi Arampatzis and Jaap Kamps. A study of query length. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 811–812. ACM, 2008.

[3] Jaime Arguello. Aggregated Search. *Foundations and Trends® in Information Retrieval*, 10(5):366–501, 2017.

[4] Jaime Arguello, Jamie Callan, and Fernando Diaz. Classification-Based Resource Selection. 2009.

[5] Jaime Arguello, Fernando Diaz, and Jamie Callan. Learning to aggregate vertical results into web search results. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 201–210. ACM, 2011.

[6] Jaime Arguello, Fernando Diaz, Jamie Callan, and Ben Carterette. A methodology for evaluating aggregated search results. In *ECIR*, volume 11, pages 141–152. Springer, 2011.

[7] Thi Truong Avrahami, Lawrence Yau, Luo Si, and Jamie Callan. The fedlemur project: Federated search in the real world. *Journal of the Association for Information Science and Technology*, 57(3):347–358, 2006.

[8] Ziv Bar-Yossef and Maxim Gurevich. Random sampling from a search engine's index. *Journal of the ACM*, 55(5):1–74, 2008.

[9] Luisa Bentivogli, Pamela Forner, Bernardo Magnini, and Emanuele Pianta. Revising the wordnet domains hierarchy: semantics, coverage and balancing. In *Proceedings of the Workshop on Multilingual Linguistic Ressources*, pages 101–108. Association for Computational Linguistics, 2004.

[10] Krishna Bharat and Andrei Broder. A technique for measuring the relative size and overlap of public Web search engines. *Www*, 30(1-7):379–388, 1998.

[11] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.

[12] Horatiu Bota, Ke Zhou, Joemon M Jose, and Mounia Lalmas. Composite retrieval of heterogeneous web search. In *Proceedings of the 23rd international conference on World wide web*, pages 119–130. ACM, 2014.

[13] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[14] Derek Bridge, Mehmet H. Goker, Lorraine McGinty, and Barry Smyth. Case-based recommender systems. *The Knowledge Engineering Review*, 20(03):315, 2005.

[15] Robin Burke. Knowledge-based recommender systems. *Encyclopedia of library and information systems*, 69(Supplement 32):175–186, 2000.

[16] James P. Callan, Zhihong Lu, and W. Bruce Croft. Searching distributed collections with inference networks. *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 21–28, 1995.

[17] Jamie Callan. Advances in Information Retrieval. *Advances in Information Retrieval*, 7:127–150, 2002.

[18] Jamie Callan and Margaret Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19(2):97–130, 2001.

[19] Jamie Callan, Margaret Connell, and Aiqun Du. Automatic discovery of language models for text databases. *SIGMOD Rec.*, 28(2):479–490, June 1999.

[20] Claudio Carpineto and Giovanni Romano. A survey of automatic query expansion in information retrieval. *ACM Computing Surveys (CSUR)*, 44(1):1, 2012.

[21] Walter Carrer-Neto, María Luisa Hernández-Alcaraz, Rafael Valencia-García, and Francisco García-Sánchez. Social knowledge-based recommender system. Application to the movies domain. *Expert Systems with Applications*, 39(12):10990–11000, 2012.

[22] Suleyman Cetintas, Luo Si, and Hao Yuan. Learning from past queries for resource selection. *Cikm*, pages 1867–1870, 2009.

[23] Aleksandr Chuklin, Anne Schuth, Katja Hofmann, Pavel Serdyukov, and Maarten De Rijke. Evaluating aggregated search using interleaving. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 669–678. ACM, 2013.

[24] Aleksandr Chuklin, Anne Schuth, Ke Zhou, and Maarten De Rijke. A Comparative Analysis of Interleaving Methods for Aggregated Search. *ACM Transactions on Information Systems*, 33(2):1–38, 2015.

[25] Charles LA Clarke, Maheedhar Kolla, Gordon V Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 659–666. ACM, 2008.

[26] Cyril Cleverdon. The cranfield tests on index language devices. In *Aslib proceedings*, volume 19, pages 173–194. MCB UP Ltd, 1967.

[27] Thomas Demeester, Dolf Trieschnigg, Ke Zhou, Dong Nguyen, and Djoerd Hiemstra. Fedweb greatest hits presenting the new test collection for federated web search. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, New York, NY, USA, 2015. ACM.

[28] Paul Dourish. What we talk about when we talk about context. *Personal and Ubiquitous Computing*, 8(1):19–30, 2004.

[29] Marina Drosou and Evaggelia Pitoura. Search result diversification. *ACM SIGMOD Record*, 39(1):41–47, 2010.

[30] Alexander Felfernig and Robin Burke. Constraint-based recommender systems: technologies and research issues. *Proceedings of the 10th international conference on Electronic commerce ICEC '08*, 8(5):1–10, 2008.

[31] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[32] Michael Granitzer, Christin Seifert, Silvia Russegger, and Klaus Tochtermann. Unfolding Cultural, Educational and Scientific long-tail content in the Web. *CEUR Workshop Proceedings*, 997(April 2017), 2013.

[33] Luis Gravano. *Querying multiple document collections across the Internet*. PhD thesis, stanford university, 1997.

[34] Matthias Hagen, Martin Potthast, Benno Stein, and Christof Bräutigam. Query segmentation revisited. In *Proceedings of the 20th international conference on World wide web*, pages 97–106. ACM, 2011.

[35] Donna Harman. Relevance feedback and other query modification techniques., 1992.

[36] Xuefeng He, Jun Yan, Jinwen Ma, Ning Liu, and Zheng Chen. Query topic detection for reformulation. In *Proceedings of the 16th international conference on World Wide Web*, pages 1187–1188. ACM, 2007.

[37] Zan Huang, Hsinchun Chen, and Daniel Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems*, 22(1):116–142, jan 2004.

[38] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[39] D Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*, volume 40. 2011.

[40] Dietmar Jannach. Fast computation of query relaxations for knowledge-based recommenders. *AI Communications*, 22(4):235–248, 2009.

[41] Arlind Kopliku, Karen Pinel-Sauvagnat, and Mohand Boughanem. Aggregated search: A new information retrieval paradigm. *ACM Computing Surveys (CSUR)*, 46(3):41, 2014.

[42] Adenike M Lam-Adesina and Gareth JF Jones. Applying summarization techniques for term selection in relevance feedback. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 1–9. ACM, 2001.

[43] F W Lancaster. Precision and recall. *Encyclopedia of Library and Information Science*, (January):2346–2351, 2003.

[44] Fabiana Lorenzi and Francesco Ricci. Case-based recommender systems: A unifying view. In *Intelligent Techniques for Web Personalization*, pages 89–113. Springer, 2005.

[45] Jie Lu and Jamie Callan. Federated search of text-based digital libraries in hierarchical peer-to-peer networks. In *Advances in Information Retrieval*, pages 52–66. Springer, 2005.

[46] Thomas R Lynam, Chris Buckley, Charles LA Clarke, and Gordon V Cormack. A multi-system analysis of document and term selection for blind feedback. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 261–269. ACM, 2004.

[47] Bernardo Magnini and Gabriela Cavaglia. Integrating subject field codes into wordnet. In *LREC*, pages 1413–1418, 2000.

[48] Stuart E. Middleton, Nigel R. Shadbolt, and David C. De Roure. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems*, 22(1):54–88, jan 2004.

[49] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[50] Jesse Montgomery, Luo Si, Jamie Callan, and David A Evans. Effect of varying number of documents in blind feedback: analysis of the 2003 nrrc ria workshop bf_numdocs experiment suite. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 476–477. ACM, 2004.

[51] Eli Pariser. *The filter bubble: What the Internet is hiding from you*. Penguin UK, 2011.

[52] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. A picture of search. In *InfoScale*, volume 152, page 1, 2006.

[53] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[54] Duc Truong Pham, Stefan S Dimov, and Chi D Nguyen. Selection of k in k-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 219(1):103–119, 2005.

[55] Filip Radlinski and Susan Dumais. Improving personalized web search using result diversification. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 691–692. ACM, 2006.

[56] Sriram Raghavan and Hector Garcia-Molina. Crawling the hidden web. Technical report, Stanford, 2000.

[57] Anand Ranganathan and Roy H. Campbell. An infrastructure for context-awareness based on first order logic. *Personal and Ubiquitous Computing*, 7(6):353–364, 2003.

[58] Bradley James Rhodes. *Just-in-time information retrieval*. PhD thesis, Massachusetts Institute of Technology, 2000.

[59] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Recommender Systems Handbook*, volume 54. 2015.

[60] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, volume 7, pages 410–420, 2007.

[61] Raoul Rubien, Hermann Ziak, and Roman Kern. Efficient search result diversification via query expansion using knowledge bases. In *Database and Expert Systems Applications (DEXA), 2015 26th International Workshop on*, pages 286–290. IEEE, 2015.

[62] James Rucker, Marcos J Polanco, W E B Page, Recommendation System, and That Uses. Recom Syst. *Communications of the ACM*, 40(3):73–75, 1997.

[63] Rodrygo LT Santos, Craig Macdonald, and Iadh Ounis. Exploiting query reformulations for web search result diversification. In *Proceedings of the 19th international conference on World wide web*, pages 881–890. ACM, 2010.

[64] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pages 85–90. IEEE, 1994.

[65] Christin Seifert, Jörg Schlötterer, and Michael Granitzer. Towards a feature-rich data set for personalized access to long-tail content. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 1031–1038. ACM, 2015.

[66] Milad Shokouhi. Central-rank-based Collection Selection in Uncooperative Distributed Information Retrieval. *Proceedings of the 29th European Conference on IR Research*, pages 160–172, 2007.

[67] Milad Shokouhi. Federated Search. *Foundations and Trends® in Information Retrieval*, 5(xx):1–102, 2011.

[68] Milad Shokouhi, Leif Azzopardi, and Paul Thomas. Effective query expansion for federated search. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 427–434. ACM, 2009.

[69] Milad Shokouhi, Falk Scholer, and Justin Zobel. Sample sizes for query probing in uncooperative distributed information retrieval. In *Frontiers of WWW Research and Development-APWeb 2006*, pages 63–75. Springer, 2006.

[70] Luo Si and Jamie Callan. Relevant document distribution estimation method for resource selection. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 298–305. ACM, 2003.

[71] Luo Si and Jamie Callan. Modeling search engine effectiveness for federated search. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 83–90. ACM, 2005.

[72] Shanu Sushmita, Hideo Joho, Mounia Lalmas, and JM Jose. Understanding domain relevance in web search. *WWW 2009 Workshop on Web Search*, pages 1–4, 2009.

[73] Shanu Sushmita, Hideo Joho, Mounia Lalmas, and Robert Villa. Factors affecting click-through behavior in aggregated search interfaces. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 519–528. ACM, 2010.

[74] Paul Thomas and David Hawking. Evaluating sampling methods for uncooperative collections. *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '07*, page 503, 2007.

[75] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[76] Günter Urak, Hermann Ziak, and Roman Kern. Do ambiguous words improve probing for federated search? In *International Conference on Theory and Practice of Digital Libraries*, pages 438–441. Springer, 2016.

[77] Ellen M Voorhees. The philosophy of information retrieval evaluation. In *CLEF*, volume 1, pages 355–370. Springer, 2001.

[78] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508. ACM, 2006.

[79] William Webber, Alistair Moffat, and Justin Zobel. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS)*, 28(4):20, 2010.

[80] Qi Ye, Feng Wang, and Bo Li. Starrysky: A practical system to track millions of high-precision query intents. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 961–966. International World Wide Web Conferences Steering Committee, 2016.

[81] Xiaoyan Yu, Fernando, and Edward a Fox. Hard Queries can be Addressed with Query Splitting Plus Stepping Stones and Pathways. *IEEE Data Eng. Bull.*, 28:29–38, 2005.

[82] Hermann Ziak and Roman Kern. Evaluation of pseudo relevance feedback techniques for cross vertical aggregated search. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 91–102. Springer, 2015.

[83] Hermann Ziak and Roman Kern. Query splitting for context-driven federated recommendations. In *Database and Expert Systems Applications (DEXA), 2016 27th International Workshop on*, pages 193–197. IEEE, 2016.

[84] Hermann Ziak and Roman Kern. Evaluation of contextualization and diversification approaches in aggregated search. In *Database and Expert Systems Applications (DEXA), 2017 28th International Workshop on*, pages 103–107. IEEE, 2017.

[85] Hermann Ziak, Andi Rexha, and Roman Kern. Know at the social book search lab 2016 mining track. In *CLEF (Working Notes)*, pages 1190–1196, 2016.