Bauhaus-Universität Weimar
Faculty of Media
Degree Program Computer Science and Media

Document Clustering with Query Constraints

Master's Thesis

Matthias Busse

1. Referee: Prof. Dr. Matthias Hagen
2. Referee: Prof. Dr. Volker Rodehorst

Adviser: Dipl.-Med.-Sys.-Wiss. Tim Gollub

Date of Submission: February 18th, 2015

Matthias Busse
Document Clustering with Query Constraints

Master's Thesis
Degree Program Computer Science and Media
Chair of Big Data Analytics, Faculty of Media
Bauhaus-Universität Weimar, Germany

Declaration of Authorship

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, February 18th, 2015

.....
Matthias Busse

Abstract

In this thesis, we revisit the document clustering problem from an information retrieval perspective that explicitly addresses the need for appropriate cluster labels.

The challenge of finding meaningful cluster labels is one of the major impediments for an extensive use of document clustering in down-market information retrieval systems. As a result of our novel perspective, we present several query constraints for both cluster label generation and cluster analysis that ensure consistency with a keyword-based reference search engine. Under the assumption that the search engine retrieves results that successfully satisfy the information needs of the users formulated as queries, the introduced query constraints guarantee document clusterings with meaningful cluster labels that allow the users to anticipate the content of a cluster from its label.

We present a flexible four-step processing pipeline for a labeled document clustering with a variety of methods for each step and place a particular focus on clustering algorithms that satisfy the query constraints. A variant of the k-means algorithm along with noun phrases as cluster labels, a search engine using a topic model and a corresponding relevance constraint for the result lists turns out to perform best. In the course of the evaluation, we introduce a novel soft clustering evaluation measure as well as an extended and publicly shared version of the popular Ambient data set. We compare the effectiveness of our novel approach with two state-of-the-art algorithms, Descriptive k-means and k-means plus chi-square. While the found clusters and their documents are of comparable high quality for each of the algorithms, the evaluation of the corresponding cluster labels reveals a great diversity in their explanatory power. In a user study with 49 participants, we show that labels generated by our approach are of significantly higher discriminative power, leading to an increased separability of the found clusters in comparison to the clusters of the baseline algorithms. With regard to the descriptive power of the cluster labels, our approach significantly outperforms one of the baseline algorithms, while achieving an equivalent descriptive power compared to the second baseline algorithm.

Contents

1 Introduction

We are facing a constantly increasing amount of electronic documents. Digital libraries and encyclopedias enlarge their collections every day, websites produce novel documents likewise, and people can easily spread personalized information through blogs and tweets. Overwhelmed with this growing and manifold textual data, the demand for techniques organizing it in manageable forms are increasing as well. Most systems maintaining a huge document collection tackle the problem by providing a search engine. If we have a concrete piece of information in mind that we want to search for, like a particular website or person, we can express our information need in form of a query and submit that query to the search engine, which in turn locates documents that possibly contain the answer to satisfy our information need. However, in some cases we might only have a vague information need which cannot be formulated by a single query. For example, imagine to create a query that answers the question: Which subjects dominate the headlines of all major newspapers today? Or you are given all the publications of an author and the task is to examine the research fields the author is engaged in. One could answer these questions by simply sifting through all available documents and explore each document manually, but such an approach is inappropriate as it requires too much time and effort. Such exploration problems also occur in combination with search engines. Users tend to formulate short and ambiguous queries that match various documents covering a variety of topics. As a consequence, several documents from the returned result list are likely to be discarded by the users for their irrelevance to their actual information needs.

In computer science, a popular approach to tackle the exploration problem is to use document clustering algorithms. The typical objective of such algorithms is to automatically organize a collection of unstructured documents into a smaller number of coherent classes (called *clusters*) in such a way that documents in one cluster are somewhat more similar to each other than to those in other clusters. Along with meaningful labels for the clusters, which summarize the content of each cluster, a topical overview is obtained and the user can easily get a general idea what the documents are about, i. e., the newspapers' headlines of today or the author's research fields. Moreover, document clustering facilitates the systematic exploration of a diversified document collection, as often returned by a search engine when submitting a query. With the presence of a descriptive label for each document cluster, users can now quickly narrow the focus to just a particular subset of the documents which really meets their information need.

In conclusion, the objectives of document clustering for the use in information retrieval are twofold: The algorithms have to (1) unveil the topical structure in a document collection and (2) provide meaningful descriptions to communicate the discovered structure

to the user. The former task, which is referred to as *cluster analysis*, has been a topic of research for a considerable long time and many studies confirm the clustering algorithms' effectiveness for finding groups of semantically related documents. However, conventional clustering algorithms such as the popular k-means are not capable of producing meaningful cluster labels. Instead, a subsequent step, called *cluster labeling* and representing the second task of document clustering described above, is performed. Cluster labeling techniques strive for examining the topic of the documents per cluster in order to not only find labels that summarize the content of every cluster but also distinguish the clusters from each other. One major drawback of common labeling techniques is their limitation in selecting only statistical features from the documents as labels; for example, they concatenate the most prominent terms occurring in the documents, called *keywords*. However, a broad list of keywords tends to represent different and unrelated aspects of the documents and will often fail to provide a readable label to the user. In other cases, an appropriate label may not occur directly in the text. For example, consider the following texts of four newspaper articles:

1. The defender stole the ball from the opponent and set up the fast break.
2. After a rude tackle of the defender in the box, the referee called a penalty.
3. The referee tried to calm the audience down when she was about to serve.
4. She was badly disappointed after losing the expensive election campaign.

Document clustering algorithms are able to discover a common concept for the first three documents, as they pairwise share a set of terms, and group these documents in the same cluster. Although it is clear to the reader that an appropriate label would be in the sense of "sports", the cluster labeling step would rather suggest a vague phrase like "defender referee" according to the terms' frequency in the cluster documents. In our opinion, the necessity of a meaningful cluster label goes hand in hand with the grouping of semantically similar documents. With the absence of a descriptive label, users are forced to investigate each cluster for its covering topic. An improvement in the document representation is not accomplished, irrespective of the clustering's quality.

To incorporate the crucial aspect of meaningful labels into the objectives of document clustering, we revisit the task of document clustering from an information retrieval perspective, and think of a user's decision for the suitability of a label as follows: The presentation of a cluster label activates a concept in the user's mind, and each document that is relevant to this concept is classified under that label. This process is conceptually very closely related to the standard task of information retrieval—query-based search. Here, a search engine is confronted with a query instead of a label, and the task of the search engine is to return only those documents that are relevant to the submitted query. That is, both the user and the search engine are faced with a textual term, and have to decide whether a document is related to that specific term or not. This analogy leads us to suggest the use of a search engine to model the user, and to think of cluster labels as search queries which have to retrieve the documents of the associated cluster.

With our way of looking at document clustering from an information retrieval perspective, we establish an explicit connection to the tremendous amount of research that has been achieved in this field, and can therefore exploit especially the wisdom of web search engines. In particular, for a topic query like "sports", the retrieval model of the search engine might also retrieve those documents that do not include the query terms, except the three relevant articles from the aforementioned example. Furthermore, the interplay between information retrieval systems and cluster analyses brings forth an intuitive approach to hierarchical search result clustering: Once the relevant aspects in a document collection are unveiled in form of search queries, each of the corresponding result sets can then serve as input for another iteration of the clustering process, which in turn leads to a new set of now more detailed aspects, i. e., search queries. For instance, after unveiling the topic "sports", the corresponding documents can be further divided in order to discover subtopics like "basketball", "soccer" and "tennis". In case that the retrieval system stores more documents than given by the user, the system can also implicitly act as a recommendation system by returning not only a subset of the given documents as cluster documents, but in addition other unknown yet relevant documents for the unveiled query; for example, a fourth article dealing with sports. Before we are able to implement these ambitious efforts, we first have to answer the following research question that is addressed by this work:

- Does a document clustering algorithm that is based on search queries sustain the high clustering quality of conventional algorithms while enhancing the explanatory power of the cluster labels?

In this thesis, our scientific contributions are threefold: (1) We present a flexible four-step processing pipeline for document clustering that revisits the clustering problem from an information retrieval perspective by using search queries. (2) In the course of the evaluation, we present an extended and publicly shared version of the Ambient data set with 4,680 manually annotated documents. (3) We conduct a user study involving 49 participants in order to compare the explanatory power of the cluster labels generated by our approach and two state-of-the-art algorithms.

The remainder of the thesis is organized as follows: Chapter 2 reviews related scientific work and discusses the extensive research regarding document clustering and labeling. We briefly describe various approaches and point out major differences to our approach. Chapter 3 starts with an introduction to the processing pipeline of our approach. We then investigate each of its steps separately for its versatile opportunities to influence the resulting labeled document clustering—with a particular focus on clustering algorithms. In Chapter 4, we evaluate our approach with respect to the novel Ambient++ data set and compare it with two state-of-the-art clustering algorithms. Finally, Chapter 5 summarizes our work and concludes with an outlook to future research.

2 Related Work

In this chapter, we review research of available algorithms and methods from the broader spectrum of labeled document clustering and which closely correspond to the ideas presented in this thesis. We describe the various approaches and point out the major differences to our approach.

Document clustering has its initial application in improving the effectiveness of document retrieval. Van Rijsbergen noticed in his cluster hypothesis that "closely associated documents tend to be relevant to the same requests" [Rij79]. The idea was to group the collection of documents beforehand, and to enrich search results with documents from clusters matched by the submitted query. In doing so, also documents not containing the query terms but being just as relevant were retrieved. Even though the applied clustering algorithms varied in the metrics used for measuring the similarity between documents, they were all hidden in the background with no intention of revealing the resulting clusters to the user. This situation changed when people realized that clusters could be used as a browsing interface in order to explore and organize a set of unstructured documents. Since conventional algorithms typically discovered clusters with respect to a mathematical model that was not explainable in textual form, a readable label for the cluster's content was required. The relevance of a cluster can then be accessed in a faster way than by examining the cluster for its documents. In this context, numerous clustering algorithms and labeling techniques have been proposed. Carpineto et al. attend to this large variety and present a classification scheme that groups the approaches into three classes: data-centric, description-aware, and description-centric [CORW09]. We adopt this scheme to provide an overview of algorithms in the field of labeled document clustering. As our idea utilizes search queries for facing the clustering problem, we put particular focus on approaches closely corresponding to this idea and emphasize them in another class—query-centric algorithms. Note that we do not consider this class as an extension to the scheme presented by Carpineto et al. but rather as a spotlighting of query-based approaches.

Data-centric Algorithms

Algorithms in this category address the problem of document clustering as merely another data clustering problem. Such algorithms are not limited to applications in the text domain as they generally focus on disciplines that involve studies of diversified data; for example, detecting objects in image segmentation. The algorithms repre-

sent the data in mathematical models, calculate similarities between the data objects, and seek for the best partitioning of these data objects. With respect to the text domain, a popular representation was introduced by Salton et al. as the Vector Space Model [SWY75]. Given a collection of documents, each document d is represented as a vector of terms $\mathbf{d} = \langle w(t_1, d),\ w(t_2, d),\ w(t_3, d),\ ...,\ w(t_n, d)\rangle$, where $t_0, t_1, ...t_n$ denotes a global set of unique terms (features) and $w(t_i, d)$ expresses the weight (importance) of term t_i to d. The weights are typically non-negative values indicating the distribution of occurrences of a term in the document (called term frequency, TF), possibly adjusted with a correcting factor that highlights the importance of a term to that particular document in the context of the entire collection (called inverse document frequency, IDF). For example, the TF vector for d with the text "A day without laughter is a day wasted" is shown in the following figure as $\mathbf{d}_{\text{TF}}$, whereas a fictional TF-IDF vector is represented as $\mathbf{d}_{\text{TF-IDF}}$:

t	d_{TF}	$\mathrm{d}_{\text{TF-IDF}}$
a	2	0.1
day	2	1.4
without	1	0.6
laughter	1	1.1
is	1	0.2
wasted	1	0.8

Note that the terms "a" and "is" are likely to appear in almost every document in the collection and therefore, the adjusted TF-IDF values are significantly lower than for salient words such as "laughter" or "wasted". For each document, the corresponding vector provides a location for the document in the term-document space and the similarity of two documents can be computed by calculating the distance between their vectors, e. g., the cosine of the angle between them.

The number of available methods of how to find clusters containing similar documents is overwhelming. Although one of the most popular algorithms, k-means, is explained in Section 3.4, the interested reader is referred to Jain and Dubes who provide a comprehensive review for cluster algorithms [JD88]. One of the pioneering examples for using clusters as a document browsing interface is the Scatter/Gather algorithm introduced by Cutting et al., which creates a cluster hierarchy in an iterative process [CKPT92]. In the Scatter phase of the two-step approach, documents are clustered by some cluster algorithm, and the resulting clustering is shown to the user. After selecting one cluster of interest, its documents are used as input for the next iteration in the Gather phase.

However, conventional cluster algorithms do not consider the generation of a label that can be presented to the user. In case a description is necessary, the problem is tackled as an independent, subsequent step of the clustering process. In its most simple

approach, labels are derived from salient words (called keywords) occurring in the cluster documents. Stein and Meyer zu Eißen sort such keywords in a global list according to their frequency and assign each keyword successively to that cluster in which it occurs proportionately at the most; thereby, preventing duplicated terms [SM04].

Further techniques resort to probability theory and determine if a term's occurrence in one cluster is statistically independent from its occurrence in other clusters [MRS08]. Pearson's chi-square test is capable of calculating this dependency and is explained in more detail when describing the baseline algorithm in Section 3.5. The generation of cluster labels can also be performed indirectly by using external knowledge, in the hope that more comprehensive labels are generated. Even though Carmel et al. also extract keywords as the previous approaches, these do not only serve as potential cluster labels but also for obtaining related Wikipedia articles. Their titles and categories extend the list of label candidates, which in turn is evaluated by several independent judges such as pointwise mutual information [CRZ09].

The output of labeling methods used for data-centric algorithms is often not satisfying. One reason is that the label generation is based on term frequencies in a cluster as a whole rather than for each cluster document. It can thus not be guaranteed that the label reflects the main content of every document, which in turn might be confusing for a user being confronted with this label. This violates our view of an appropriate cluster label as presented in the introduction in Chapter 1. Another shortcoming is that most approaches generate labels that form a sequence of probably unrelated keywords and often lack understandability for the user. In conclusion, the performance of the preceding clustering step is irrelevant if the subsequent cluster labeling step cannot sustain the given quality. This fact suggests to close the gap between both steps and reveals the second class of cluster algorithms.

Description-aware Algorithms

The main issue of the above described data-centric algorithms is that they try to generate a label from a mathematical model of text that is not intended for this purpose. Description-aware algorithms are, as the name suggests, aware of this labeling problem and therefore try to ensure that the construction of cluster labels produces results that are comprehensive and meaningful to the user.

One way to achieve this goal is to use algorithms that assign documents to clusters based on a single feature (called monothetic clustering). In case this feature is selected carefully, i. e., a user considers it as something meaningful and precise, it can be used to sufficiently describe the cluster and separate it from the others. This idea first appeared in the Suffix Tree Clustering (STC) algorithm published by Zamir and Etzioni [ZE98]. They use frequently recurring phrases as similarity features between documents. Assuming that documents covering related topics should also use a similar vocabulary, frequent phrases can be used to identify those documents and assign them to a common

cluster. The algorithm works in two phases: First, it discovers groups of documents that share a single frequent phrase (base clusters) by utilizing suffix trees. Second, it iteratively merges base clusters whose phrases overlap to a certain extent, and this process is repeated to form the final output clusters. However, the merging step is based on the single-linkage criterion, which tends to result in chain-merging of base clusters that should not be combined. As a result, the combined phrases of merged clusters, which form the cluster labels, might be unrelated and therefore misleading for the user.

A follow-up algorithm is SnakeT by Ferragina and Gulli [FG04]. The algorithm uses non-contiguous phrases as features, which the authors call approximate sentences. Although the criterion for forming a cluster is still the fact of sharing a sufficient number of phrases (here, approximate sentences), the authors go one step further and enrich the potential set of cluster labels with phrases acquired from a predefined ontology. This way, they try to achieve labels that are more meaningful to users.

But still, the cluster analysis precedes and dominates the labeling task—something the next class of algorithms tries to balance and sometimes even reverse.

Description-centric Algorithms

Members of this group consider the cluster labels as the crucial elements of the overall clustering. They assume that if a cluster cannot be described by a meaningful label, it is probably of no value to the user and should be removed entirely. The quality of the description hence precedes the allocation of the document to the cluster.

The focus on significant labels was especially substantiated with the rise of the World Wide Web. Search engines have proven to be an invaluable tool for locating pages of interest, and sifting through ranked lists of search results is still a popular way to browse the Web. But users tend to submit ambiguous queries and to click only on top-ranked result pages; therefore, they might miss relevant information in the tail [JGP+05, ABDR06]. Such practices demand the organization of the search results in a manageable form. Algorithms that are specifically designed for clustering search results are widespread, and we will only present a small selection. Carpineto et al. provide a detailed overview [CORW09].

Lingo is one of the pioneering examples that are particularly suited to solve the problem of search result clustering [OSW04]. After preprocessing the snippets (text fragments of a web page shown in a search engine's result list) in a first step, which includes stemming and stopword marking, Osiński et al. determine frequent terms and phrases. The authors then use Singular Value Decomposition (SVD) on the term-document matrix to extract orthogonal vectors, which are assumed to represent distinct topics in the input snippets. In the final step, Lingo assigns the documents to the extracted topic clusters by calculating the relevance of a document to each topic using the Vector Space Model. Lingo is part of Carrot2, an open source framework for search result clustering.[1]

[1] `http://project.carrot2.org`, Last accessed: February 13th, 2015

Weiss revisits conventional data-centric algorithms and adjusts the popular k-means algorithm to a description-centric pattern [Wei06]. First, the algorithm discovers groups of similar documents by running the k-means algorithm on the document collection with TF-IDF values as features. The topics covered by each group can be obtained from the centroids of the resulting clusters. It then extracts frequent phrases as potential cluster labels. After representing both the label candidates and the cluster centroids using the Vector Space Model (VSM), the algorithm determines those phrases that are most similar to the centroids. In the document assignment phase, Weiss searches for cluster documents that are relevant to such a topic phrase, again by utilizing the VSM.

Scaiella et al. propose a graph-based approach to cluster search snippets [SFMC12]. Therefore, every snippet is annotated with a set of topics, which are represented by means of Wikipedia articles. A bipartite graph structure is then built where nodes are either snippets or topics, and weighted edges determine the semantic relatedness between topic-to-topic or snippet-to-topic edges. The final snippet clusters are obtained by performing an iterative graph-cut algorithm, and are labeled with the corresponding Wikipedia title.

Zeng et al. tackle the cluster labeling problem by replacing the unsupervised clustering with a supervised classification [ZHC+04]. After creating a set of salient phrases from the input snippets, each of these label candidates is then weighted by aggregating several factors, e. g., phrase frequency, length, entropy, and phrase independence. The specific weights for each of these factors were learned from examples of manually prioritized cluster labels. After assigning the documents to relevant salient phrases, the final clusters are generated by merging overlapping candidate clusters.

Description-centric algorithms correspond well with our view at the cluster labeling problem by placing their focus on label quality instead of cluster quality. However, the approaches have several shortcomings. The labels for topics veiled in the document collection are drawn from salient words or phrases that are spread in the documents, assuming that the extracted terms are appropriate to describe the topics, although it depends on the degree of abstraction a user strives for. Even if the label is appropriate, only documents that share such a label to a certain extent are merged in one cluster. Yet documents not containing this topic label but being just as relevant are not considered to belong to the topic's cluster. Furthermore, it is difficult to evaluate the proper assignment of a document to a topic that is exclusively based on the presence of the topic label in this document.

As stated in the introduction in Chapter 1, we believe that search queries are able to overcome these drawbacks, given the extensive research especially on search engines for the Web. Relevant approaches are discussed in the last group of algorithms.

Query-centric Algorithms

Search queries are well-known to users from their daily web search experience. Cluster algorithms utilizing search queries as labels try to take advantage of this familiarity, assuming that the linkage of result documents to a submitted query is similar to the linkage of cluster documents to a given label.

Both approaches of Osiński et al. [OSW04] and Weiss [Wei06] utilize search queries in their algorithms as well, although they are categorized in the group of description-centric algorithms described in the previous section. This is based on the fact that the authors only use queries for validating the clustering in the last step of their approaches, i. e., the cluster labeling step. Given the keyword-based generated queries as labels, they submitted each of them to a search engine in order to obtain relevant cluster documents. In fact, we consider search queries as the driving force of the cluster analysis: Their existence precedes the formation of clusters.

So far, salient phrases occurring in a document are considered to be appropriate for describing the content of the document and thus for serving as features forming the base of the cluster analysis. Gollub et al. propose a modern means of document content descriptors—keyqueries [GHMS13]. Their idea is that those queries returning a given document in their top ranks when submitted to a reference search engine describe the document content well enough to stand out from the indexed collection. Since the concept of keyqueries is one of the inspiring ideas for this thesis, we will depict keyqueries in more detail in Section 3.4. Fuhr et al. suggest that queries which retrieve a particular document are in general suitable to represent a document [FLSG12]. In their optimum clustering framework (OCF), they use the relevance scores or retrieval ranks for each query as feature weights to find clusters of similar documents. However, they do not address the problem of labeling the resulting clusters.

Bonchi et al. utilize queries in order to determine different topics in a document collection [BCDG08]. Given a query, they decompose it with the help of a query log into a set of queries, where the union of their result sets covers the documents retrieved from the given query. The authors assume that each result set represents a different topic in the collection. Although we strive for the generation of queries as well, our approach starts from a given document collection instead of an initial query.

The same holds for the approach of Wang et al. [WZ07]. They also utilize a search query log to retrieve syntactically similar queries to a given query. Assuming that the discovered queries denote different aspects of the query, the authors group them by running a star clustering algorithm with TF-IDF as features, and label each cluster with a representative query, i. e., with the one that is nearest to the centroid of the cluster. Once the aspects of the query are learned, each of its result documents is categorized in the aspect that it is most similar to. Again, our goal is not to decompose a given query but rather a given document collection, and we do not resort to a query log.

Gollub et al. presented a similar approach to ours. They utilize their aforementioned keyqueries to construct a hierarchical classification system for digital libraries [GVHS14]. The first step in the framework is the computation of keyqueries for each document in the given library's collection, and this computation is restricted by five constraints and influences, for example, the syntax of a keyquery. In order to represent the first level in the hierarchical taxonomy, the authors strive for a set of keyqueries which covers all the given documents and their unveiled topics. This task is stated as an optimization problem that maximizes the recall with respect to the documents, and is tackled with a greedy set-cover algorithm. Once the first level of suitable keyqueries is computed and the user wants to explore a specific topic, this process is repeated iteratively for the subset of documents that are retrieved for that topical keyquery.

In this thesis, we seize the approach of our research group members Gollub et al. and continue their work. In particular, we want to exploit the strength of the retrieval system in a higher degree and strive for further cluster algorithms that are especially designed for search queries.

3 Document Clustering with Query Constraints

The following chapter describes our novel approach towards a labeled clustering of documents with the help of search queries. While Section 3.1 motivates and overviews our idea, the subsequent sections depict the processing pipeline in detail: First, the generation of meaningful cluster label candidates is explained in Section 3.2. The candidates are then processed by a search engine in Section 3.3 to serve as features for a cluster analysis that is discussed in Section 3.4.

3.1 Processing Pipeline

The challenge of finding meaningful cluster labels is one of the major impediments for an extensive use of document clustering in down-market information retrieval systems. Our goal is to formulate a document clustering objective that addresses the user's need to anticipate the content of a cluster from its label, and that allows us to validate a cluster labeling in terms of this need. The following sections reveal the problem of cluster labels, introduce our idea to overcome this problem and further propose constraints to not only find meaningful cluster labels but also to group similar documents in the same cluster.

3.1.1 The Cluster Labeling Problem

In order to demonstrate the impediments to a meaningful cluster labeling, we first focus on the typical steps involved in a document cluster analysis, which are depicted in Figure 3.1. The mere presentation of a given document collection D to a user U is not feasible, since it takes too much time and effort to sift through all documents for getting a topical overview of the collection. Document cluster analysis addresses this problem by detouring the direct connection. Therefore, each document of D is represented under a uniform model, which is constructed by selecting discriminative features of the documents (e. g., term frequencies), and which leads to a set of document representations R. A clustering algorithm can then analyze the documents for similarities due to their common representation, and find a clustering C that groups documents with high similarities and distinguishes those with low similarities. In order to establish an improved connection between the document subsets in C and the user U, the last step

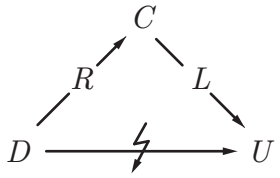

Figure 3.1: Typical steps involved in a document cluster analysis. Since the direct connection between a given document collection D and a user U is not feasible, the cluster analysis represents each document under a uniform model and stores the representations in a set R, merges similar documents with the help of the representations to a clustering C, and entitles the clusters with meaningful labels L, which are then presented to the user.

seeks for meaningful cluster labels L that allow users to anticipate the topics shared among the documents in the clusters.

While the connection between D and C is very robust due to the vast amount of different clustering algorithms for various applications, the major drawback arises in the connection between C and U. Note that the cluster labels are responsible for the quality of this connection. The worse the labels presented to the user, the weaker the connection, and the lower the probability for the user to derive the topic shared in the cluster documents. Once this crucial connection is broken, an improvement in the document representation cannot be accomplished, irrespective of the clustering's quality. That is, the connection then is not superior to the direct connection between D and U. In order to evaluate the quality of cluster labels, Stein and Meyer zu Eißen depict three main characteristics of meaningful labels [SM04]:

Comprehensive	The syntax of the label should be appropriate.
Descriptive	The label should speak for each document in a cluster.
Discriminative	The semantic overlap between two cluster labels should be minimal.

One of the major drawbacks of conventional cluster labeling techniques is that the descriptive power of the labels is not addressed in a sufficient manner, as they try to establish a connection between the labels L and the users U without involving the users themselves. Assuming that concatenations of salient terms in the cluster documents are meaningful enough to describe the clusters, the algorithms do not validate the actual appropriateness of the label to the users.

In conclusion, the quality of cluster labels depends on multiple factors, which in turn makes it difficult to evaluate their suitability. The problem is even more challenging if there are different users with different knowledge that might require different labels for the same documents. While a user being unfamiliar with a topic seeks for a rather general label providing a broad overview, this particular description may not be satisfying for

an experienced user who desires a more detailed label. So, whether a cluster label serves the purpose of describing the topic of a cluster depends also on the individual user that inspects the cluster label. For our work, we define a cluster labeling $L = \{l_1, ..., l_k\}$ for k clusters to be valid for a user u if u would group the cluster documents c_i under cluster label l_i in a manual effort when provided with L and D.

3.1.2 From Cluster Labels to Search Queries

For assessing the validity of a cluster labeling without manual inspection, we introduce the concept of a user model μ that strives for modeling the human classifier u. In order to derive a user model for the cluster labeling task, we think of the user's decision making for a valid label as follows: The presentation of a cluster label activates a concept in the user's mind, and each document that is relevant to this concept is classified under that label. This process is conceptually very closely related to the standard task of information retrieval—query-based search. Here, a search engine is confronted with a query instead of a label, and the task of the search engine is to return only those documents that are relevant to the submitted query. That is, both the user in the labeling task and the search engine in the retrieval task are faced with a textual term and have to decide whether a document is related to that specific term or not. This analogy leads us to propose the use of a search engine to model the user, and to think of cluster labels as search queries that have to retrieve the documents of the associated cluster. In doing so, it is straightforward to validate the descriptive power of a cluster label. That is, to determine whether the search engine retrieves the same documents for a cluster label that a user would classify under the same label.

The assumed analogy implies that the language used to formulate search queries and the language used to construct conventional cluster labels are equal and interchangeable. In this regard, both search queries as well as cluster labels are abstract and minimalistic linguistic expressions. In classification systems like the Dewey Decimal Classification, which we consider as gold standards for cluster labels, classes are in most cases described through noun phrases (e. g., information retrieval), or conjunctions of these (e. g., digital libraries and archives). For search queries, a recent study revealed that 70 % are strict noun phrases, and another 8 % can be reordered to noun phrases (e. g., algorithm clustering), which in turn denotes that the vast majority of search queries fulfills the requirement of being an appropriate cluster label [BJR08]. Even though search queries are generally used with less diligence, we think of the search query language as a slang of the class label language. Moreover, the concept of search queries and their association to relevant search results is well understood today due to the user's daily web search experience.

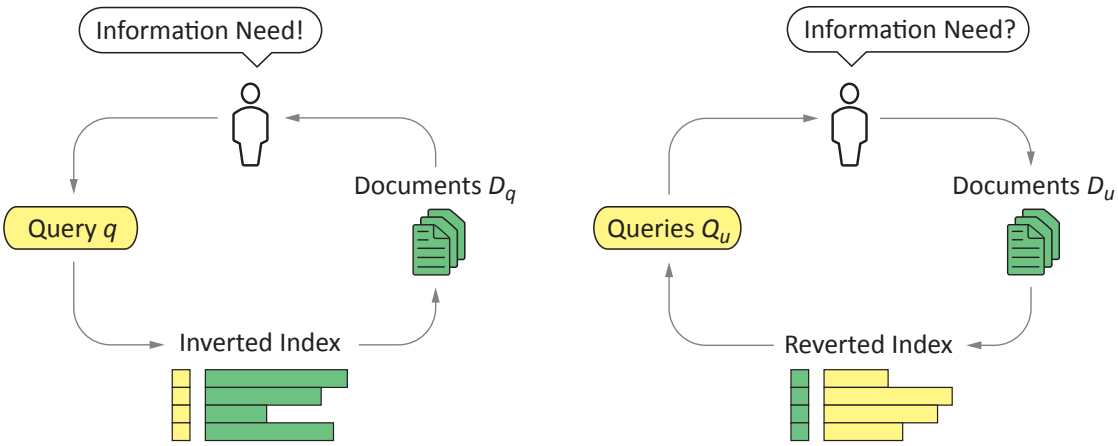

Figure 3.2: From information retrieval to document clustering. The task of the latter is depicted in the right loop and turns out to be the reverse problem of the former, which is shown in the left loop.

3.1.3 Introducing Query Constraints

Equipped with the definition of a valid cluster labeling and the use of search queries as suitable cluster labels, the task of document clustering can be formulated as follows: Given a set of documents and a reference search engine, we see the task of document clustering to find a set of search queries whose result sets group similar documents and separate dissimilar ones. In this respect, document clustering turns out as the reverse problem of query-based search, i. e., the standard scenario in information retrieval. The interaction sequence between a user u and the retrieval system is depicted in the left loop in Figure 3.2. After u states her information need in form of a search query q, the retrieval system resorts to its inverted index and responds with a list of relevant documents D_q, which are typically ranked with respect to their aggregated similarity to each query term. As aforementioned, this scenario is an appropriate analogy to a user expecting conceptually relevant documents when being confronted with a cluster label.

In contrast, the task of document clustering is shown in the right loop of Figure 3.2. Here, u is equipped with a document set D_u and wants to learn about the information needs that could be satisfied with this document set. The clustering system can account for this need by providing a set of search queries Q_u, each of it reflecting one specific aspect hidden within the document collection. The data structure needed for finding discriminative queries is a reverted index that stores for every document the queries retrieving it. Along with their retrieved documents as cluster contents, the returned queries then form the labeled document clustering of D_u. This assumption implies that the potential clusters of a document are given by the queries for which it is retrieved.

For the cluster analysis, this leads to a constraint within the constrained clustering terminology [BDW08], which we call common-query constraint CQ:

CQ Two documents *cannot link* if they do not share a common search query.

That is, only if documents are retrieved for one and the same query, they are assumed to be similar and are thus candidates for a shared cluster. By introducing the common-query constraint, the discriminative power of the cluster labels is guaranteed. Each label must discriminate its cluster documents from other clusters, because similar documents would have been retrieved for that labeling query as well and therefore grouped in the same cluster.

In conclusion, this section introduced two concepts: (1) the search engine that models the user to retrieve relevant documents for a query and (2) the common-query constraint to find clusters of similar documents. While the former validates the descriptive power of a cluster label, the latter validates its discriminative power. However, both steps require the presence of a cluster label, i. e., the actual generation of potential labels has to precede. Once an appropriate vocabulary for the labels is determined, the comprehensive power (the third property of meaningful cluster labels) can be validated.

Figure 3.3 illustrates how we incorporate the vocabulary generation, the search engine, and the cluster analysis in our approach. Note that this figure will also serve as the guideline for the remainder of this chapter. After extracting a meaningful vocabulary from the documents (Section 3.2), we index the documents by means of this vocabulary using a search engine (Section 3.3). In order to find similar documents, we build a reverted index, which represents each document by its retrieving queries. With queries as features for the documents, a clustering algorithm can now utilize the reverted index and find a clustering that satisfies the common-query constraint (Section 3.4) and for which the selected search queries serve as cluster labels. The next sections explain various approaches for each step in detail, and introduce further query constraints to achieve a good clustering with meaningful labels.

3.2 Vocabulary Generation

In order to label the clusters, it is necessary to build a vocabulary from which the labels arise. The entries of the vocabulary form the potential cluster labels, which also serve as search queries to retrieve relevant documents, i. e., the cluster documents.

The vocabulary generation is an important step in the processing pipeline since the choice of vocabulary terms determines the comprehensive power of the cluster labels. In case the terms are ambiguous, not comprehensive, or too specific, the cluster labels will inevitably also exhibit such problems and will fail to reflect the content of the clus-

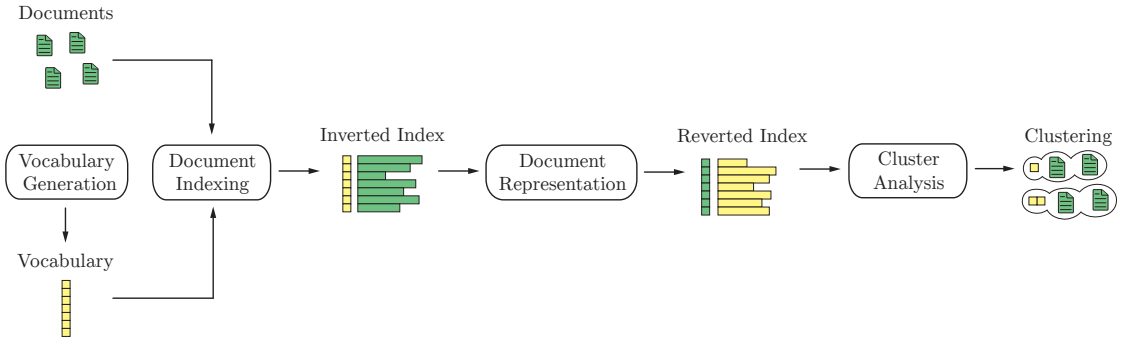

Figure 3.3: The steps involved in our document clustering pipeline. After extracting a meaningful vocabulary, we model the user by indexing the documents using a search engine. In order to find similar documents, we represent each document by its retrieving queries in a reverted index, which is then utilized by the cluster analysis in order to find a clustering, for which selected search queries serve as cluster labels.

ter to the user. Moreover, the size of the vocabulary has a deep impact on the overall performance of our approach. A small vocabulary limits the amount of queries being formulated and submitted to the search engine. In an ideal situation, each hidden cluster can be unveiled by exactly one query, which retrieves only the cluster documents. When it comes to the automatic generation of cluster labels, such a desired scenario is not feasible. On the one hand, some queries are likely to be too general and cover more than only the cluster's topic; thus, the retrieved results also contain documents of other clusters. On the other hand, some queries might be too specific and the corresponding topic is not discussed in every cluster document; thus, less results are retrieved. Consequently, a large vocabulary expands the range of cluster labels and can establish connections between documents that are not covered by one single query but by many others. That is, if a document a is related to a document b, and b is in turn related to another document c, then a is also related to c (transitive relation). The cluster analysis can then exploit such relations and group all three documents in one cluster.

With respect to the syntax of cluster labels, category names in classification systems or Wikipedia are considered to be ideal [SM04, TC06, Wei06]. They describe a collection of documents in a meaningful and informative manner; therefore, good cluster labels should apply such human description patterns, and a user should not detect any differences between a human and a generated cluster label. As already mentioned in Section 3.1.2, category names turn out to be made of noun phrases or conjunctions of these; therefore, we consider noun phrases as suitable to serve as cluster labels. For readability reasons, we restrict the number of conjunctions to one, so that a cluster label reads like "NOUN PHRASE and NOUN PHRASE"; for example, "Digital Libraries and Archives".

In conclusion, according to the notion of the common-query constraint presented in Section 3.1.3, we refer to this restrictions as the query-syntax constraint QS:

QS A cluster label consists of noun phrases or a conjunction of these.

In the remainder of this section, we describe two approaches to generate an appropriate vocabulary out of noun phrases.

3.2.1 Natural Language Processing

Since the late 1950's, natural language processing (NLP) explores the interactions between computers and human (i. e., natural) languages, for which part-of-speech tagging (POS tagging) is one of the essential components. That is, marking up a word in a text as corresponding to a particular part of speech, based on both its definition and its context, which is the relationship with adjacent and related words in a phrase, sentence, or paragraph. In a simplified form, POS tagging refers to the identification of words as nouns, verbs, adjectives or adverbs. Among others, one challenge is that many words, especially common ones, can serve as multiple parts of speech. For example, "book" can be both a noun ("the book on the table") or verb ("to book a flight"), while "set" can even be a noun, verb or adjective. The interested reader is referred to Manning and Schütze [MS99], who provide a comprehensive introduction to the field of NLP.

The tagging of words in a text with their corresponding part of speech facilitates the identification of noun phrases, which typically take the following form (not all elements need to be present):

DETERMINER + PRE-MODIFIERS + NOUN + POST-MODIFIERS

A noun phrase is formed with a noun as the head word and some other elements that modify it and are therefore the head's dependents. Table 3.1 lists and exemplifies the possible parts of speech for each element in the structure of noun phrases.
As an example of identified noun phrases, consider the following sentence:

While having lunch, one of the college students expresses her desire to travel widely.

Having marked each word with its part of speech and applying the typical form of noun phrases, we can extract three noun phrases, which are emphasized by underlines.
Although these phrases form grammatically correct noun phrases, they do not inevitably serve as good candidates for cluster labels in their present form. For instance, the determiner adds no relevant information to the noun phrase and is rather an unnecessary expansion (*one of the* and *her*). The same holds for the post-modifiers, which are not mandatory to complete the meaning of the phrase, even though giving extra or specific information about the noun. For our work, we consider noun phrases to be only a

Table 3.1: The grammatical elements of a noun phrase. Each element is listed with its corresponding part of speech and an example.

Element	Part of Speech	Examples
DETERMINER	Articles	*the*, *a[n]*
	Demonstratives	*that*, *which*
	Possessives	*my*, *whose*
	Quantifiers	*all*, *many*
	Numerals	*one*, *two*
PRE-MODIFIERS	Adjectives	*yellow*
	Adjective phrases	*really lovely*
	Noun adjuncts	*college* in *the college student*
POST-MODIFIERS	Prepositional phrase	*of Berlin*
	Relative clauses	*which we saw yesterday*
	Participial phrases	*sitting on the beach*
	Infinitive phrases	*to travel widely* after *desire*
	Dependent clauses	*that the world is round* after *fact*

concatenation of pre-modifiers and a head noun. Consequently, the extracted noun phrases for the above example are:

While having lunch, one of the college students expresses her desire to travel widely.

Since the pre-modifiers are not yet restricted in their length, arbitrarily long cluster labels can be generated. This violates our understanding of appropriate labels and therefore, we investigate the Wikipedia for the length distribution of its category names. Categories allow articles to be placed in one or more groups, and allow those groups to be further categorized. The analysis depicted in Table 3.2 shows that the category names in Wikipedia have an average length of 3.87 terms. As a consequence, we allow the extracted noun phrases to consist of maximal four terms. According to the query-syntax constraint, a cluster label might be a conjunction of two noun phrases, leading to a maximum length of a cluster label of eight terms (plus the term "and"). We refer to this restriction as the query-length constraint QL for cluster labels:

QL A cluster label consists of maximum four terms per at most two noun phrases.

Rationale for the noun phrase extraction is to select only those phrases which maximize the chance of retrieving similar documents with regards to the content. Therefore, the objective is to extract phrases that represent the main content of a document, which are called *keyphrases*. As already stated in the introduction of this section, a carefully

Table 3.2: Length distribution of category names in Wikipedia.

Terms	1	2	3	4	5	6	≥ 7
Categories	83,878	316,299	459,501	423,977	282,899	143,638	175,817
Ratio	4.45 %	16.77 %	24.36 %	22.48 %	15.00 %	7.62 %	9.32 %

chosen vocabulary reduces the overall costs of using the search engine (i. e., the number of queries submitted to the search engine). In order to facilitate a beneficial pruning of extracted candidates, a ranking function is necessary. Barker and Cornacchia describe a simple system for assessing the value of noun phrases, based on their length, their frequency of occurrence, and the frequency of their head noun in the document [BC00]. In a user study, the authors demonstrate the capability of their approach, and show that it performs approximately as well as a state-of-the-art, corpus-trained keyphrase extractor. We apply the approach of Barker and Cornacchia in our work and implement it using the Apache OpenNLP library[1] for POS tagging. With a ranking of noun phrases at hand, we can limit the extracted phrases of a document to the most meaningful ones by determining the number of noun phrases to extract, which then serve as search queries and potential cluster labels. Section 4.2 investigates different settings for this number in detail and as it turns out, six noun phrases per document are suitable to form a vocabulary that is appropriate in length and quality. Additional extracted noun phrases expand the vocabulary unnecessarily, without producing further valuable cluster labels.

However, the automatic identification of noun phrases reveals several drawbacks for the use as cluster labels. For instance, consider the following two extracted noun phrases: (1) “biggest theaters” and (2) “bigger theatres”. On the one hand, they represent the same topic, even if differently spelled, and should therefore not occur twice in the otherwise unnecessary large vocabulary. Even if only one of the phrases is selected, it is not appropriate as a label anyway for its inflected form. So, on the other hand, the extracted noun phrases should be listed in their canonical forms; that is, as they appear in dictionaries or classification systems. We address the two problems with lemmatization, which is an algorithmic process in computational linguistics for determining the lemma (canonical word) for a given word. With lemmatization, we can both identify the above phrases as one and the same, and generate phrases in an appropriate grammatical form (here, “big theater”). Again, we utilize the Apache OpenNLP library[1] to find the lemmatized form for each noun phrase. Another drawback besides inflated forms of phrases is the fact that the automatic extraction of noun phrases might produce meaningless noun phrases like “big issue” or “common example”, and include them in the vocabulary list. The approach presented in the next section tries to overcome this problem by only extracting keyphrases contained in a predefined vocabulary.

[1] `http://opennlp.apache.org`, Last accessed: February 13th, 2015

3.2.2 Wikipedia Article Titles

The objective of a keyphrase extraction for cluster labels is to capture the main content of a document in brief phrases. Then, documents concerning the same topic can be retrieved by the search engine when confronted with such phrases. Since the automatic identification of noun phrases can lead to inappropriate candidates, it seems worthwhile to find keyphrases from a controlled and predefined vocabulary, which only consists of well-formed and suitable phrases.

In order to follow the approach being presented in this section, we narrow the focus on the Wikipedia's Manual of Style,[2] which shall help editors in writing articles with consistent and precise language, layout, and formatting. In particular, the guidelines regarding the linking of words and phrases to other Wikipedia articles are of special interest. The main recommendations of that guidelines are as follows: (1) Authors should provide links to articles with relevant connections to the subject and that facilitate a deeper understanding. This includes people, events, technical terms and other topics. (2) Everyday words understood by most readers in the context as well as terms being unrelated to the subject in question should not be linked. (3) The authors should take special care to the proper amount of linked articles. An overlinked article contains an excessive number of links, making it difficult to identify links that improve the reader's understanding significantly.

In their work, Mihalcea and Csomai indicate that the criteria for selecting linked words or phrases in Wikipedia articles appear to be very similar to those used for extracting keyphrases in a document [MC07]. Since the links should be limited to phrases that have a corresponding Wikipedia article, they suggest to construct a vocabulary out of keyphrases that contains only the Wikipedia article titles. For this purpose, we apply the query-length constraint for cluster labels as introduced in the previous section, and select only those titles with a maximum length of four terms. In addition, we discard Wikipedia article titles that solely consist of one or more stopwords, dates, special or non-latin characters, because they usually do not serve as meaningful cluster labels. As of November 2014, the English Wikipedia comprises 11,344,987 article titles including disambiguation pages and redirects, and after pruning according to the aforementioned restrictions, 2,869,974 titles remain and form the predefined vocabulary. Similar to the previous approach, we apply a lemmatization to the vocabulary terms and the document text in order to find every relevant match.

However, an arbitrary document is likely to contain plenty of Wikipedia article titles, and it is advisable to extract only those titles that represent the main aspects in the document. As already stated in the previous section, such an intent demands a ranking function. In this respect, Mihalcea and Csomai introduced a measure called keyphraseness [MC07]. They estimate the probability of a phrase p to be selected as a keyphrase by counting the number of documents where the phrase appears as a link to

[2] `http://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style`, Last accessed: February 13th, 2015

other Wikipedia articles (D_{p_link}), divided by the total number of documents where the phrase appeared (D_p):

$$keyphraseness(p) = \frac{count(D_{p\_link})}{count(D_p)}$$

We apply the idea of Mihalcea and Csomai, and calculate for each Wikipedia article title its keyphraseness. After extracting the titles in a document, we rank this list according to the titles' keyphraseness and are able to limit the extraction to the most valuable titles.

To conclude, this section presented two approaches to extract a vocabulary with meaningful entries as potential cluster labels. As the evaluation will reveal in Section 4.2, we will use the extraction of noun phrases in our approach. The next step is to index the documents with respect to this vocabulary and therefore, to model the user in the decision of whether a document is relevant to a vocabulary term or not.

3.3 User Modeling

The user modeling step constitutes the document indexing step in our document clustering pipeline. Its objective is to model the user reading a cluster label, so that the user's decision whether the label is appropriate for a given document can also be made by the model. This allows a valid assignment of relevant documents to each of the generated vocabulary terms in the previous step without manual decisions of the user.

As already stated in Section 3.1.2, we consider a search engine to be an appropriate and versatile model of the user. We can therefore exploit especially the wisdom of web search engines, which also aims at presenting the user only those documents that the user is likely to consider as relevant to the submitted query. The core of every search engine is its retrieval model. Given a query, it resorts to the already indexed documents and determines which documents match the query and which do not. Therefore, the retrieval model can be seen as the modeled user.

The BOOLEAN model is the classical and basic retrieval model used in information systems. It is based on Boolean logic and classical set theory in that both the documents to be searched and the user's query are conceived as sets of terms. The matching of the query and the documents is determined by whether or not the documents contain the query terms, according to the logical operator. While the AND operator implies the presence of each query term in a requested document, the OR operator only assumes the presence of one of the query terms. For the use as cluster labels, it is not feasible to concatenate terms using the OR operator, since such labels might be confusing to a user. For example, a user confronted with the label "bass guitar" might not expect documents about the fish that only contain the term "bass".

However, the BOOLEAN model reveals several disadvantages for modeling the user. For instance, the binary decision criteria for the query terms in the documents may retrieve too few documents, as not all relevant documents are likely to contain each term. Moreover, the mere presence of terms makes it difficult to rank the output, but some documents might be more relevant to the query than others. Finally, all query terms are equally weighted. But general terms might not be as important for a document as very specific topical terms, which only appear in few, but all the more relevant, documents.

In the remainder of this section, we describe three retrieval models that address different drawbacks of the BOOLEAN model and are thus more appropriate to model the user in our application scenario.

3.3.1 Vector Space Modeling

In order to retrieve relevant documents for a given query, the search engine utilizes an algebraic model for representing every document d and the query q as vectors. Each dimension of such a vector corresponds to a separate term t_i. If t_i occurs in the document or query respectively, its value w in the vector is non-zero. That is, d and q are represented as vectors

$$\mathbf{d} = \langle w(t_1, d),\ w(t_2, d),\ w(t_3, d),\ ...,\ w(t_n, d)\rangle,$$
$$\mathbf{q} = \langle w(t_1, q),\ w(t_2, q),\ w(t_3, q),\ ...,\ w(t_n, q)\rangle.$$

Several different ways of computing the values, also known as term weights, have been developed. One of the best known schemes is TF-IDF weighting. Note that this model was already mentioned in Chapter 2. Since it is compared with the other retrieval models represented in this section in the experimental part of this thesis (Chapter 4), it is discussed here for the sake of completeness again in more detail.

The TF-IDF weighting scheme consists of two factors. The first, TF, is the term frequency. Rationale for this factor is that documents which contain a term more often are generally more relevant to a query, i. e., the intention of the user. The more frequent a term occurs in a document, the greater its score. It is common to use the square root of the term frequency as a damping function to limit the influence of a large number of term occurrences in a single document. The second factor, IDF, is the inverse document frequency, which measures the amount of information a term provides, i. e., whether the term is common or rare across all documents. The greater the occurrence of a term in different documents, the lower its score. The score is the logarithmically scaled fraction of the total number of documents in the collection D by the documents that contain the term n_t. If the term t is not in the collection D, this will lead to a division-by-zero for the IDF factor. It is therefore common to adjust the denominator by adding 1. Altogether, the TF-IDF weight of a term t for a single document d is

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \cdot \text{IDF}(t) = \sqrt{freq(t, d)} \cdot \log\left(\frac{N}{n_t + 1}\right).$$

The individual weight of a term for each document brings forth the ability to score and rank a document's relevance with regard to a given query. A sophisticated and widely used ranking function is the comparison of the deviation of the angles between the vector of each document d and the query q. In practice, it is easier to calculate the cosine of the angle between the vectors, instead of the angle itself:

$$\cos \phi(q, d) = \frac{\mathbf{q} \cdot \mathbf{d}}{\|\mathbf{q}\| \cdot \|\mathbf{d}\|},$$

where $\mathbf{q} \cdot \mathbf{d}$ is the dot product of the document and the query vectors, $\|\mathbf{q}\|$ is the norm of vector $\mathbf{q}$, and $\|\mathbf{d}\|$ is the norm of vector $\mathbf{d}$.

In comparison to the BOOLEAN model, it is no more a binary decision whether a document is relevant to a query or not. The cosine similarity allows the computation of a continuous degree of relevance between queries and documents. For example, consider two documents that both contain the query terms. It is likely that one document satisfies the information need of the user more than the other, which might only mention the query terms briefly in a minor paragraph. However, the BOOLEAN model is forced to retrieve both documents, which might not correspond to a real user's decision. With the ranking of documents according to their relevance score, we are able to trim the result list and therefore, ensure that a query only retrieves those documents which are *about* the query topic, i. e., the documents which are returned in the top results of a query. In doing so, we increase the descriptive power of a query, i. e., a cluster label. Nevertheless, determining the position in a query result list where the mere occurrence of the query phrases turns into "aboutness" is not trivial. To facilitate a binary decision, we suggest two different relevance constraints:

Top-k Constraint The top-k constraint decides upon the relevance of a document to a query, if the document is retrieved in the top-k ranks of the search result list. Any documents appearing beyond these k first documents are discarded and not considered as relevant to the query. In our approach, we set $k = 10$, as this is a popular cut-off for many evaluation measures in information retrieval (e. g., precision@10). Such a restriction can rise problems if a possible document occurring on rank $k + 1$ is as relevant as the first k, but is however not included in the retrieved documents. Moreover, remember that the documents retrieved for a query also serve as the cluster documents. The determination of the parameter k thus limits the maximum number of documents that a cluster can consist of. In case the approximate size of the clusters is known in advance, such problems are negligible.

Score Constraint The score constraint decides upon the relevance of a document to a query, if the retrieval score of the document exceeds a certain threshold. This threshold is determined for every single query and therefore facilitates a variable range of relevant documents, in contrast to the fix-sized range of the top-k constraint. However, the determination of an appropriate threshold is challenging. As

a starting point, one could suggest the average score of the retrieved documents. Since it is likely that many documents will have some even very small relevance to the query, the average tends to be very small, leading to an excessive amount of relevant documents. We therefore strive for a threshold that is independent of the number of retrieved documents, and which evolves from the simple average formula as follows:

$$threshold = \frac{\sum s_i}{N} = \sum \frac{1}{N} \cdot s_i = \sum \frac{s_i}{\sum s_i} \cdot s_i = \frac{\sum {s_i}^2}{\sum s_i},$$

where s_i denotes the score for each retrieved document and N denotes the number of relevant documents. Rationale for this formula is the consideration of the normalization factor $\frac{1}{N}$ as a contribution of each document to the average score, and that this contribution can be interpreted as the ratio of the document score to the sum of all scores, that is $\frac{s_i}{\sum s_i}$.

Beside the increased descriptive power of a query due to the relevance constraint, the vector space model has another advantage over the BOOLEAN model; that is, the vector space model allows partial matching of the query terms. In other words, not all terms have to occur in a document, but also a single term leads to a certain relevance of the document for the query. In case that the other terms are very specific or the document only contains the corresponding synonyms, the user might nevertheless consider the document as relevant, and the user model can thus benefit from partial matching. This advantage might conflict with the issue mentioned for the OR operator in the BOOLEAN model (irrelevant documents in the result list for just containing a subset of the query terms). However, the cosine similarity in combination with the relevance constraint addresses this issue. The fewer terms intersecting between the query and the document, the wider the angle between their two vectors, and the smaller the cosine similarity, i. e., the relevance score. By applying the relevance constraint, such a document might be discarded anyway, depending on the importance of the intersecting terms.

3.3.2 Probabilistic Modeling

The TF-IDF weighting scheme is a term scoring function, which can be incorporated in a document scoring method using a similarity measure (e. g., cosine similarity). However, the notion of the mere term intersection between a query and a document does not translate directly into an assessment of whether the document is a "good" document to give to the user or not. It is rather a question of probability, since the search engine has only an uncertain understanding of the user and thus, makes an uncertain guess of whether a document satisfies the query or not. Probability theory provides a principled foundation for such reasoning under uncertainty, and probabilistic retrieval models exploit this foundation to estimate how likely it is that a document is relevant to a query. The most popular implementation of probabilistic models, and besides TF-IDF one of the

most successful retrieval algorithms, is BM25. Since the theoretical basis of BM25 goes beyond the scope of this thesis, we refer the interested reader to a thorough summary provided by Robertson and Zaragoza [RZ09]. In the thesis at hand, we describe the practical differences between TF-IDF and BM25.

Both TF-IDF and BM25 define a weight for each term as a product of a TF function and a IDF function, and then process these weights to compute the score for the whole document with respect to the given query. Moreover, both models acknowledge that for highly frequent terms, a further increase in term frequency has little impact on the relevance. However, the saturation function of BM25 asymptotically approaches a limit for high term frequencies, while the square root of TF-IDF has no boundary. This difference in growth implies that with an increasing number of a term, a document gets a higher relative increase in the term frequency score for TF-IDF than for BM25.

Another major difference between both models is the use of the document length in BM25. By considering the document length, BM25 tries to compensate for the fact that a longer document has in general more terms and is thus more likely to have a higher term frequency without necessarily being more pertinent to the term and therefore, no more relevant to the query. Yet some longer documents actually have a wider scope, what justifies their longer texts. BM25 is sensitive to this issue by adjusting the term frequency using a document-length normalization factor that is calculated from two tuning parameters k_1 and b, the document length $|d|$, and the average document length $avgdl$ in the collection. The overall term weight is then calculated by multiplying the term frequency score with a similar adjustment as used in TF-IDF, the inverse document frequency IDF, which measures the amount of information a term provides, i. e., whether the term is common or rare across all documents. In order to get the score of a document d for a query q, BM25 adds up the weights for every query term t_i as follows:

$$score_{\text{BM25}}(q,d) = \sum_{i=1}^{n} \text{IDF}(t_i) \cdot \frac{freq(t_i,d) \ \cdot \ (k_1+1)}{freq(t_i,d) \ + \ k_1 \cdot \left(1 - b \ + \ b \cdot \frac{|d|}{avgdl}\right)},$$

where k_1 and b are usually chosen as $k_1 \in [1.2, 2.0]$ and $b = 0.75$ ($0 \leq b \leq 1$). That is, setting $b = 1$ will perform full document-length normalization, while $b = 0$ will switch normalization off. The IDF weight is computed slightly differently than for TF-IDF:

$$\text{IDF}(t) = \log \frac{N - n_t \ + \ 0.5}{n_t \ + \ 0.5},$$

where N is the total number of documents in the collection, and n_t is the number of documents containing the term t.

Note that the user model only benefits from BM25 if we establish one of the relevance constraints that were already introduced for TF-IDF, i. e., the determination of a top-k or a score threshold to determine the most relevant documents retrieved by a query.

Otherwise, every document with a similarity measure greater than zero would also be part of the cluster.
Although both models apparently have computational similarities, BM25 has a substantive mathematical foundation (again, see the summary of Robertson and Zaragoza for more details [RZ09]) and therefore, a higher reputation in the research field of information retrieval. However, both the models TF-IDF and BM25 suffer from a problem known as *vocabulary mismatch*, i. e., the problem that the vocabulary of a query and the vocabulary of relevant documents can differ substantially, since people may choose different words to describe the same thing. The next section both describes the cause and effect in more detail and presents a retrieval model that is able to overcome the vocabulary mismatch problem.

3.3.3 Topic Modeling

Human words are characterized by a great diversity. Users in different contexts or with different needs, knowledge, or linguistic habits will describe the same information using various terms. Examples are the use of synonyms, abbreviations, or different spellings (e. g., "theater" versus "theatre"). In a comprehensive user study, Furnas et al. reveal that different people (experts in the same field) will name the same thing differently in 80 % of the data [FLGD87]. With respect to query-based search, term-matching retrieval models like TF-IDF and BM25 will inevitably fail to cover all relevant documents but only those documents that share a common vocabulary with the query. For considering solely the mere occurrence of words in both query and documents, these retrieval models are called *bag-of-words* models.

However, a user submitting a query expects all relevant documents in the result list, or expects all relevant documents in a cluster when reading the corresponding cluster label, respectively. We therefore strive for a retrieval model that not only considers the syntax of a query but also its topical semantics and thus alleviates the vocabulary mismatch problem. In this regard, so-called *topic* models can be integrated into the retrieval process and bring forth the semantic retrieval of documents. The objective of topic models is to discover the abstract topics that occur in a collection of documents, and the rationale for that task is as follows. Intuitively, a document dealing with a particular topic is likely to contain particular terms more or less frequently: "goal" and "offside" will appear more often in documents about soccer, whereas "racket" and "serve" will appear in documents about tennis, and "ball" and "player" will appear equally in both. A document typically concerns multiple topics in different proportions; thus, a document that is 90 % about soccer and 10 % about tennis would probably contain about 9 times more soccer terms than tennis terms. A topic model captures this intuition in a mathematical model, and allows to discover the topics in a set of documents and what each document's balance of topics is, based on the statistics of the terms in each document. A crucial point for the discovery of topics (also called *concepts*) is the knowledge base

where the topics arise from. It is either derived implicitly from the document collection, as in Latent Semantic Indexing [DDF+90], or given explicitly by external resources. One approach in the latter direction that has attracted a lot of attention in recent years is the Explicit Semantic Analysis (ESA) model by Gabrilovich and Markovitch [GM07]. In the remainder of this section, we briefly describe ESA and consider it as the third alternative for the user modeling step.

In essence, ESA represents a document with respect to the Wikipedia article space (the external "concepts explicitly defined and described by humans" [GM07]), and can then calculate the similarity between two documents by comparing their abstract representations. Formally stated: Given an input document d, ESA maps it to a high-dimensional vector space, where each dimension corresponds to an article a_i from a Wikipedia database $W_k = \{a_1, ..., a_n\}$.

$$\mathbf{d} = \langle w(a_1, d), ..., w(a_n, d) \rangle,$$

where $w(a_i, d)$ expresses a weight for the *strength of association* between d and a_i. Based on a function $\mathrm{f_{as}}$ that defines the strength of association between Wikipedia articles and terms, the weight can be computed as the sum of the association strengths of all terms of $d = \langle t_1, ..., t_k \rangle$ to the article a_i:

$$w(a_i, d) = \sum_{j=1}^{k} \mathrm{f_{as}}(t_j, a_i).$$

Gabrilovich and Markovitch define the function for the association strength $\mathrm{f_{as}}$ to be the TF-IDF weight presented in Section 3.3.1, and we adapt their approach for its simple and straightforward implementation. The association strength of term t_j to article a_i is then equal to the TF-IDF value of t_j in a_i:

$$\mathrm{f_{as}}(t_j, a_i) = \text{TF-IDF}(t_j, a_i).$$

In conclusion, the association strength function indicates how strongly a given term in a document is associated to a specific Wikipedia article, and the aggregation of the strengths for each term in a document expresses the association strength for the document to that specific Wikipedia article. These values then build up the dimensions of the document representation vector (the ESA vector), which corresponds to a ranking of the Wikipedia articles according to their relevance to the document.

With the presence of ESA vectors, we can assess the similarity between a query and a document by considering the query as another (tiny) document, and computing the similarity between their ESA vectors using the cosine similarity, which was already discussed in Section 3.3.1.

A crucial aspect for the retrieval performance of ESA is the selection of concepts. As originally stated by Gabrilovich and Markovitch, ESA operates under the assumption

that the knowledge base contains topically orthogonal (unrelated) concepts. However, it was later shown by Anderka and Stein that ESA performs comparably well when it is based on a collection of newswire articles or even randomly built concept articles, which do not satisfy the orthogonality property [AS09]. Nevertheless, the findings of Anderka and Stein are based on a single and very small collection of 50 homogeneous documents. We strive for a robust framework that is capable of processing arbitrary documents (e. g., heterogeneous HTML documents), and we are especially concerned with computing document similarities with very tiny documents (i. e., queries) at hand. According to our query-length constraint (cf. Section 3.2.1), queries have a maximum length of eight terms. The small number leads to a high sensibility of the query terms' occurrences in a concept. For instance, remember the example of a query "bass guitar" in Section 3.3, and two concept articles "bass (fish)" and "bass (music)". While the former concept article might be very large with numerous occurrences of the term "bass", the latter might be rather short and thus, mentions "bass" only marginally. Although the association strength function can address this unequal distribution by normalizing the term weights (thereby, trying to adjust the term's importance in both concept articles), it cannot account for an excessive use of that term. Consequently, the query will not only activate the concept about the guitar but also the concept about the fish with a rather high similarity. Since a document about the fish will also be very similar to the concept of "bass (fish)", the query will probably retrieve that document in its top results. We apply two means to counteract that issue: (1) The problem primarily occurs when submitting queries containing ambiguous terms ("bass"), which appear in topically different documents (about the fish and about the guitar). As a consequence, we damp the influence of such an ambiguous term in a query by multiplying the association strength f_{as} with the IDF value of the term in the document collection, which will be rather small for its frequent occurrence in many documents. (2) The articles of Wikipedia are known to be of arbitrary length and therefore, address the topic of the articles to different extents. While longer articles also cover the wider scope of the topic and make references to rather general subjects, short articles tend to be very specific and only consider the directly related subjects, if any. Therefore, using long articles as concepts might lead to similarities to a number of documents in the collection, while short articles might be activated from only few documents. To examine the influence of the article length, we examine two different content extraction methods for Wikipedia articles:

Full Article This method uses the full article as a concept, which includes not only the paragraphs with textual content but also the (rather general) subsequent sections, e. g., "See also", "References", "Further Reading", "External Links" and the category table.

Extended Lead Section This method accounts for concepts of equal lengths by only considering the lead section as content. This section summarizes the article in a few

paragraphs and only makes highly relevant references to other articles. From those articles, we extract the very first paragraph and add it to the concept's content.

As Section 4.3 will reveal in the experimental part of this thesis, the extraction of the full article content leads to the best retrieval performance for ESA.

Compared to bag-of-words models like TF-IDF and BM25, the ESA model has a crucial advantage: Since a word can be usually associated to many articles and thus vector dimensions (with different weights), ESA alleviates the vocabulary mismatch problem arising in the bag-of-words models, where every word corresponds to exactly one dimension. Therefore, a query and a document can be semantically related without sharing any term, but being similar to the same Wikipedia articles. Especially for rather general terms, which are likely to constitute good cluster labels for their summarizing nature, this is an important property, since they are known to explicitly appear in only few relevant documents [ST09]. In this regard, note that the relevance constraint (cf. Section 3.3.1) is of even higher importance to ESA than to bag-of-words models. Since ESA is able to discover even the smallest relevance between a query and a document, a variety of documents are likely to get a similarity score greater than zero. As a consequence, a weak relevance constraint leads to an exceeding result list, i. e., cluster size.

All three presented retrieval models, TF-IDF, BM25 and ESA, constitute different approaches to model the user's decision of whether a document is relevant to a given cluster label or not. After extracting cluster label candidates with high comprehensive power in Section 3.2, each candidate is submitted as a query to a search engine, which then returns all relevant documents according to the underlying retrieval model and stores the query along with its result list in an inverted index. In addition, all possible conjunctions of these pairs are stored in the inverted index by simply determining the intersection of the result list of two queries. The retrieval model determines the descriptive power of a cluster label, as it is responsible for the connection between a query and its result documents, or between a cluster label and its cluster documents, respectively. As the evaluation in Section 4.3 will reveal, we use the ESA model in our approach.

The objective of the last property of a good cluster label, its discriminative power, is the setting of semantic boundaries to other cluster labels, and which is the subject of the next and last step in the processing pipeline.

3.4 Query-constrained Clustering

According to the common-query constraint introduced in Section 3.1.3, a cluster cannot contain documents that do not share a common search query. Although the query-document pairs in the inverted index already satisfy this constraint, their amount is not only equal to the huge size of the vocabulary, but they also partly contain very large

or small clusters, depending on the descriptive power of the query. Moreover, several clusters will probably describe the same topic with different labels; thus, leading to a high redundancy. The objective of this section is to trim the list of clusters such that the discriminative power between them is maximized. In other words: The intra-cluster similarity should be high, whereas the inter-cluster similarity should be low.

If two documents a and b are retrieved for the same query, they are considered to be similar for the topic described by the query. The degree of similarity increases with the number of shared queries. So if a and another document c are retrieved by two queries, a is more similar to c than to b. Pickens et al. present a data structure that facilitates the efficient access to the queries of a document—the reverted index [PCG10]. In the reverted index, each document serves as the key for a list that contains all queries for which the document is retrieved, that is, for which it is relevant. As its name suggests, the reverted index is related to the inverted index used for search engines, but it stores queries in the *postlists* of documents—instead of documents in the postlists of queries. Based on the notion of keyphrases, Gollub et al. propose the name *keyqueries* for queries in such a postlist, and define them as: Given a document, its keyqueries are those queries returning the document in their top result ranks [GHMS13]. In this regard, our relevance constraint determines which documents are among these top results. In the remainder of this thesis, we adapt the notion of keyqueries by Gollub et al. The postlists of the documents in the reverted index serve as the document features for the cluster analysis. In the following, we describe three different cluster algorithms that satisfy the common-query constraint.

3.4.1 Set Cover Clustering

The first algorithm tackles the document clustering problem as a set covering problem (SCP), a classical question in combinatorics. It is asking if, given a set of elements D (called the universe) and a set S of n subsets of D, there is a union of n or less subsets $s_j \in S$ whose union equals the universe. For example, consider the universe $D = \{1, 2, 3, 4, 5\}$ and the set of subsets $S = \{\{1, 2, 3\}, \{2, 4\}, \{3, 4\}, \{4, 5\}\}$. Obviously, the union of S equals D. However, we can also cover all the elements of the universe with the following, smaller number of sets: $\{\{1, 2, 3\},\ \{4, 5\}\}$. Stated as an optimization problem, we strive for covering D with the smallest number of subsets s_j.

In our algorithm, we find an approximate solution to the optimization problem using a variant of the greedy set cover algorithm presented by Vazirani [Vaz01]. The documents to be clustered constitute the universe D, and the subsets s_j are formed by the documents in each postlist in the inverted index. In its original form, the greedy algorithm operates as follows: For up to k iterations, it selects that query q from Q, whose postlist (1) has a size within a certain range and (2) covers the maximum number of documents not yet covered by previous queries, and adds it to the subclasses Q_c, which constitutes the set of covering queries. This process is repeated until all documents are covered, no

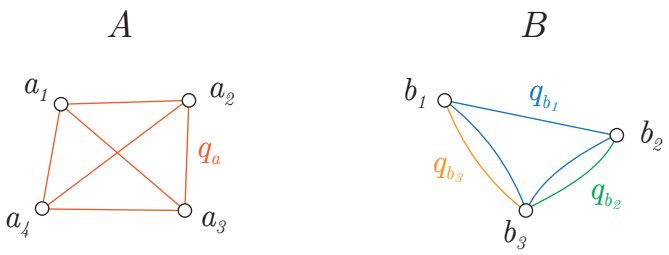

Figure 3.4: Illustration of calculating the positive rate. While cluster A is supported by only one query leading to six connections and a positive rate of 1, cluster B is supported by three queries leading to five connections and a positive rate of $1,\overline{6}$.

more candidate queries are available, or the maximum number k of iterations is reached. However, we modify this approach in that we not simply select the query which covers the most yet uncovered documents but which retrieves the most similar documents according to a novel similarity measure—the *positive rate*. The next paragraph explains the positive rate in more detail.

Positive Rate The retrieved documents of a query will later serve as documents for the cluster, and since the inter-similarity of such cluster documents should be maximized, we introduce a novel measure to account for this demand. Rationale for the measure is the following assumption: the more connections between the documents in a cluster are established by queries, the higher its intra-similarity, i. e., the positive rate. A connection between two or more documents is established whenever there is a query that retrieves these documents. For example, consider the two clusters A and B in Figure 3.4. Cluster A contains four documents a_1, a_2, a_3 and a_4. Each of the documents is retrieved by a single query q_a, which leads to six connections between them ($\binom{4}{2} = 6$). Note that this is the minimum number of connections for this amount of documents according to the common-query constraint. Another cluster B contains three documents b_1, b_2 and b_3, which are retrieved by a query q_{b_1}. Moreover, document b_2 and b_3 share another query q_{b_2}, and documents b_1 and b_3 are retrieved by a query q_{b_3}. In conclusion, cluster B is supported by three queries, which establish five connections ($\binom{3}{2} + \binom{2}{2} + \binom{2}{2} = 5$). So, while the connections in cluster A do not exceed the minimum, leading to a positive rate of $\frac{\textit{actual connections}}{\textit{minimal connections}} = \frac{6}{6} = 1$, cluster B has a positive rate of $\frac{5}{3} = 1.\overline{6}$.

In its first iteration, the greedy algorithm selects that query q whose documents have the maximum positive rate, and adds it to the covering queries Q_c. Since the documents of q are now covered, one could argue that these should not occur in another cluster, i. e., no query retrieving the documents again should be selected in the following iterations (such procedure is called *strict partitioning* clustering). In our opinion, a document might be relevant to more than one topic hidden in the document collection.

For instance, a document about "goal-line technology" is not only relevant to "sports" but also to "technology". We account for such *overlapping* clustering by discarding the connections established with documents that are already covered by a query in Q_c. So, if document b_1 in Figure 3.4 is already in another cluster, the positive rate of cluster B would decrease to $\frac{2}{3} = 0.\overline{6}$, since the three connections to b_1 would be discarded. In case that this number constitutes the maximum of all remaining query-documents pairs, this pair is selected and document b_1 would occur twice in the resulting clustering, e. g., under the cluster label "sports" as well as "technology".

When the algorithm stops due to one of the aforementioned criteria, the so far chosen queries Q_c along with the documents in their result lists form the final labeled clusters. One of the major advantages of this approach is its performance. The inverted index allows for an efficient structure to access the retrieving documents of a query and therefore the fast covering of yet uncovered documents.

3.4.2 Agglomerative Clustering

The algorithm presented in this section follows the idea of hierarchical clustering algorithms that try to build a hierarchy of clusters. A common strategy for hierarchical clustering is the agglomerative strategy, which starts with each document in its own cluster, and then merges pairs of clusters as one moves up the hierarchy. This strategy is also known as a "bottom-up" approach. Note that we do not strive for a clustering with hierarchical structures but a flat clustering with one level of clusters. Nevertheless, we exploit the valuable research achieved in the field of hierarchical agglomerative clustering and only select that level in the hierarchy whose cluster number satisfies our need. As an illustrative example, consider the dendrogram in Figure 3.5. In the first iteration, the clusters A and B (both containing a single document) are merged to a cluster AB, resulting in altogether five clusters. Following the hierarchy two more iterations, the clusters CD and E are merged. The clustering in this step of the hierarchy now consists of three clusters, namely AB, CDE and F. Although two more iterations might be feasible (shown in gray), the number of desired clusters for our clustering (here, three clusters) might be reached and therefore, the algorithm stops at this step.

In order to let the algorithm decide which two clusters should be merged in the next step, it is necessary to determine a measure that allows for evaluating and comparing the different merging possibilities. More precisely, this task is achieved by (1) the use of an appropriate measure of the distance between pairs of documents, and (2) a linkage criterion which specifies the similarity of clusters as a function of the pairwise distances of documents in the clusters. As a distance measure between two documents we suggest the number of shared keyqueries. This idea is based on the assumption that the more queries retrieve both documents, the higher the likelihood that they describe the same topic, and thus, the more similar the documents. Note that the distance measure is reversed, i. e., the smaller the distance between two documents, the higher their

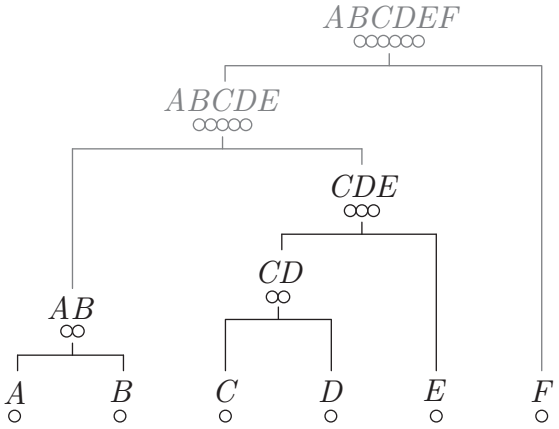

Figure 3.5: Structure of a hierarchical agglomerative clustering. The algorithm starts with each document in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. In case the desired number of clusters is reached, the algorithm stops, although more merging steps are feasible (shown in gray).

similarity. Using the distance measure between a pair of documents, we can then determine the similarity between two different clusters and their documents with respect to a linkage criterion. In common approaches, the link between two clusters is made by a single document pair, namely those two documents (one in each cluster) that are closest (i. e., smallest distance, called *single-linkage*) or farthest (i. e., longest distance, called *complete-linkage*) to each other. Both approaches strive for minimizing the corresponding extremum in order to find the next clusters to merge. Complete-linkage clustering avoids the chaining phenomenon of single-linkage clustering, where clusters might be merged due to single documents being close to each other, even though many of the documents in each cluster may be very distant to each other. We therefore use the complete-linkage criterion; in other words, our algorithm merges those two clusters, whose document pairs share a maximum of keyqueries. In case that the maximum number is shared by more than two clusters, the algorithm decides upon the ratio of shared to non-shared queries (queries that only retrieve one of the two documents) of the relevant document pair.

When merging two clusters, the documents of the new cluster inevitably share a certain amount of keyqueries, otherwise their similarity measure would be zero. However, these documents are not necessarily the only documents that are retrieved by the shared keyqueries. For example, consider the first iteration from the aforementioned dendrogram, which is highlighted in Figure 3.6. Cluster A consists of the single document a, which is retrieved by the queries q_1, q_2 and q_3, whereas cluster B with its single document b is retrieved by the queries q_2, q_3 and q_4. In case that their shared keyqueries q_2 and q_3 constitute the maximum of all document pairs, their clusters are merged to cluster AB with

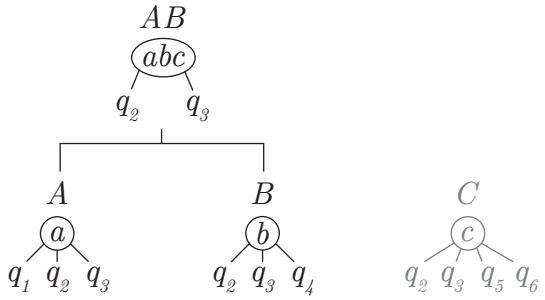

Figure 3.6: Discarding of a specialized cluster. Since the queries q_2 and q_3 of the merged cluster AB also retrieve the documents in cluster C, the latter cluster forms a specialization of AB and is therefore discarded from the clustering.

the shared keyqueries as its representing keyqueries for the next iteration. Nevertheless, cluster C with its document c is also retrieved for the queries q_2 and q_3, among others. In other words, cluster C constitutes a specialization of cluster AB. Assuming that the merging of the clusters AB and C is not performed until the desired cluster size is reached, the clustering would consist of the more general cluster AB (with a, b *and* c as cluster documents in respect of the search engine) and the specialized cluster C, whose documents (here, c) are also relevant to the query of cluster AB. In order to overcome this unintended situation, we discard all such specializing clusters in each iteration of the hierarchy construction.

When the successive merge of clusters and the decreasing amount of clusters finally lead to the desired number of clusters, the algorithm stops at this level of the hierarchy construction. Every node in the hierarchy constitutes a set of queries, each of them retrieving the documents at the leafs of the node's branch as cluster documents (e. g., queries q_2 and q_3 each retrieve the documents a, b and c in Figure 3.6). Simply concatenating the set of queries as the corresponding cluster label would in many cases violate our query-length constraint; that is, when more than two queries are left in one of the top nodes (that are, cluster AB, CDE and F in Figure 3.5). We therefore strive for the query or pair of queries from the set that best covers the cluster documents. Since all queries find at least the cluster documents, we simply choose the query (pair) that retrieves the minimum of additional documents. For example, if query q_1 solely retrieves the cluster documents a, b and c, whereas query q_2 in addition retrieves another document d, the algorithm selects q_1 to be the cluster label.

3.4.3 Constrained *k*-means Clustering

The algorithm in this section strives for answering the question if we can adopt a popular data-centric algorithm that retains the original quality of the clustering and improves the

descriptive power of the cluster labels by adhering to the common-query constraint. We decide upon the widely recognized k-means algorithm, which is not only highly scalable with a running time linear to the size of the input [LA99], but is also used by many researchers as a benchmark, so it is easy to cross-compare results with others.
The k-means algorithm uses an iterative refinement technique. Given a collection of data points, it operates in three steps (the first paragraph in the description of each step constitutes a generalized formulation, whereas the subsequent paragraph explains the application in our algorithm):

1. **Initialization** Choose k random values within the data domain as initial cluster representatives, which are called centroids.

   In our approach, the set of all keyqueries in the reverted index can be interpreted as a vector, and each document can be represented through this vector with a value of 1 for a specific entry if the document is retrieved by the query of that entry or 0 if not. For the initialization, we randomly generate k such vectors (with entries within the data domain of 0 and 1) to serve as starting centroids.

2. **Assignment** Each data point is assigned to its nearest centroid and therefore, clusters of data points are formed.

   The algorithm calculates for each document vector the dot product to all centroid vectors and assigns the document to the centroid with the highest dot product.

3. **Update** Calculate the new cluster representatives to be the centroids of the data points in the new clusters and continue with the assignment step, i. e., the k-means algorithm alternates between the two last steps.

   After all documents are assigned to their nearest centroids, the centroids are recalculated according to these assignments. Therefore, our algorithm selects that query (or query pair with respect to the query length constraint) from the inverted index whose result documents best cover the assigned documents of every centroid, and sets the new centroid as the mean vector of the result documents of that best query. In particular, we strive for the query with that result list which has the highest F-measure with regard to the assigned centroid documents. The F-measure is a well-known and often applied measure of a classification's accuracy, and is briefly explained in the following paragraph. When the maximizing query is determined, the documents in its result list represent the updated centroid by averaging their vector representations (cf. the initialization step).

F-measure When confronted with a binary classification of retrieved instances with respect to actually relevant instances, four possible situations can occur: (1) a retrieved instance is relevant, called a true positive tp, (2) a retrieved instance is irrelevant, called a false positive fp, (3) an actual relevant instance is not retrieved, called a false negative fn, or (4) an actual irrelevant instance is not retrieved, called a true negative tn.

That is, the first and last classification are correct, while the others are wrong. We can count the occurrences of each situation and then calculate both the precision and the recall of the classification. While precision is the fraction of retrieved instances that are relevant $\left(\frac{tp}{tp+fp}\right)$, recall is the fraction of relevant instances that are retrieved $\left(\frac{tp}{tp+fn}\right)$. The F-measure is a measure that combines precision and recall as their harmonic mean:

$$F_\beta = (1+\beta^2) \cdot \frac{precision \ \cdot \ recall}{(\beta^2 \ \cdot \ precision) \ + \ recall},$$

where positive values for β differently weight precision and recall. With an F_1 score, both are evenly weighted. Two other commonly used F-measures are the F_2 score, which weights recall higher than precision, and the $F_{0.5}$ score, which puts more emphasis on precision than recall.

With regard to our algorithm, the assigned centroid documents denote the relevant instances, and the documents in the query's result list are the retrieved instances. We decide to set $\beta = 1$, since we do not prefer either precision or recall. As an example of calculating the F-measure for Constrained k-means, consider the following scenario: Given 10 documents assigned to a centroid, and a query retrieving only 3 of these 10 documents, but in addition 2 other irrelevant documents. Then, the F-measure of that query is:

$$F_1 = 2 \cdot \frac{0.6 \ \cdot \ 0.3}{(1 \ \cdot \ 0.6) \ + \ 0.3} = 0.4.$$

Assuming that the query constitutes the query with the maximum F-measure, all 5 documents in its result list (3 relevant, 2 irrelevant) represent the updated centroid by averaging their vector representations (cf. the initialization step).
An illustrative example of Constrained k-means is shown in Figure 3.7. Given the desired number of clusters, $k = 3$, the algorithm randomly generates three centroids within the data domain, represented as colored rectangles (1). Each document is then assigned to the centroid that has the highest dot product with the document (2). In the next step, the algorithm resorts to the inverted index and selects the queries that best cover the assigned documents. While queries q_1 and q_2 exactly cover the assignments (leading to an F-measure of 1.0), query q_3 retrieves not only the three blue documents, but also one additional green document (leading to an F-measure of 0.86) (3). The retrieved documents for each best query form the new documents of the cluster, and the centroid is updated by averaging the vector representatives of the documents (4). The updated centroids lead to an altered assignment of the documents, since the dot products of the documents to the centroids change as well (5). The next step is again to select those queries that best cover the new assignments, and to update the centroids again. As soon as the re-assignments of the documents to the updated centroids do not change, the

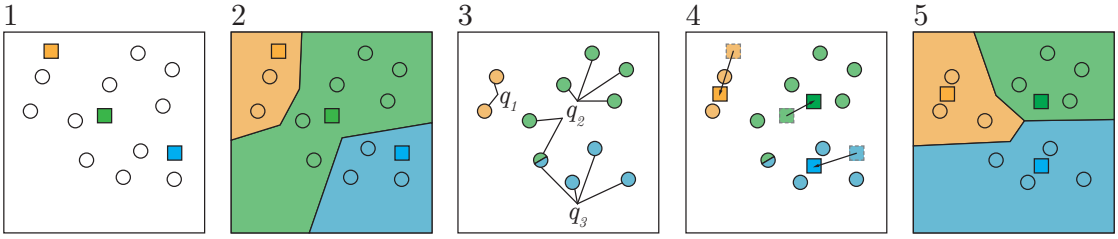

Figure 3.7: Demonstration of the Constrained k-means algorithm (here, $k = 3$). After randomly generating three centroids (1), the documents are assigned to their nearest centroids, i. e., the highest dot product (2). The queries that best cover the assignments are then determined (3), and the centroids are updated according to the result lists of the best queries (4). Since the assignments of the documents to the updated centroids has changed (5), the centroids will be again updated according to the next best covering queries in the subsequent step.

algorithm has converged and stops. All queries and corresponding result lists that best cover the documents of the centroids to this step in the alternation denote the cluster labels and corresponding cluster documents.

3.5 Baseline Algorithms

Whenever confronted with a novel approach, the question is how well it performs compared to existing approaches. In this section, we introduce two baseline algorithms that we utilize in Chapter 4 to evaluate our approach.

k-means plus chi-square

The first baseline algorithm constitutes a representative of data-centric algorithms, one of the four groups of cluster labeling techniques presented and described in Chapter 2. For a detailed explanation of data-centric algorithms, we refer the reader to that chapter. In the section at hand, we briefly recapitulate the labeling process and explain implementation details.

Similar to all data-centric algorithms, this baseline algorithm works in two subsequent steps. Given a collection of documents, it first groups similar documents by applying a conventional data clustering technique. Therefore, the algorithm represents the documents in a uniform model with discriminative features. We apply a tokenization of the document's text into unigrams and bigrams, and use their TF-IDF values as features for a k-means clustering (cf. Section 3.4.3). Unigrams and bigrams are special cases of so-called n-grams, which are contiguous sequences of n items from a text (where $n = 1$ is called a unigram, $n = 2$ is called a bigram). We use words as such items, so that

Table 3.3: Example distribution of a term t in a cluster C, which consists of documents d.

	$d \in C$	$d \notin C$
$t \in d$	$N_{11} = 70$	$N_{10} = 100$
$t \notin d$	$N_{01} = 30$	$N_{00} = 300$

the example text "a rose is a rose" is tokenized into the unigrams "a", "rose" and "is", as well as the bigrams "a rose", "rose is" and "is a". The k-means algorithm groups the documents in k unlabeled clusters. The second step of the baseline algorithm is to generate a label for each cluster. It therefore utilizes the chi-square test, which is a method applied in statistics to test the independence of two events [Pea00]. In particular, two events A and B are defined to be independent if $P(AB) = P(A) \cdot P(B)$ or, equivalently, $P(A|B) = P(A)$ and $P(B|A) = P(B)$. The higher the calculated value of the chi-square test, the higher the probability of a dependence between the events A and B. For the cluster labeling task, the two events are the occurrence of a term in the clustering and the occurrence of this term in the cluster. For calculating the corresponding value of the chi-square test, it is necessary to determine the occurrence of a term t in and outside of a cluster C. Table 3.3 lists an example of such a distribution for a collection of 500 documents. The dependence between the term t and cluster C and therefore the value of the chi-square test (also denoted as χ^2) is then calculated as follows:

$$\chi^2 = \frac{(N_{11} + N_{10} + N_{01} + N_{00}) \cdot (N_{11}N_{00} - N_{10} \cdot N_{01})^2}{(N_{11} + N_{01}) \cdot (N_{11} + N_{10}) \cdot (N_{10} + N_{00}) \cdot (N_{01} + N_{00})} = 72.19.$$

According to Manning et al., a value greater 10.83 denotes a statistically significant dependence between t and C [MRS08]. Since the term in the example exceeds this threshold, it is listed as a candidate term for the cluster label. The chi-square test is applied to all terms in the clustering (that is, all unigrams and bigrams), and the resulting list of candidate terms is then trimmed to those m terms with the highest chi-square test value. The value of m depends on the desired terms for a cluster label and has to be manually determined. We set m equal to the rounded average of the cluster labels of our approach, which is 3.18. In conclusion, the cluster documents found by the k-means algorithm along with the m most significant terms as cluster label constitute the labeled document clustering of this baseline algorithm.

Descriptive *k*-means

The second baseline algorithm is classified into the description-centric algorithms described in Chapter 2. Note that the Descriptive k-means algorithm [Wei06] was already

briefly explained in the previous chapter, but is depicted in the section at hand in more detail along with implementation details.

Descriptive k-means operates in four steps. (1) Given a collection, it first runs a conventional k-means algorithm with the words of the documents as features. Regarding the weighting formula for the terms, the author states that the choice is marginally important, since all investigated formulas (e. g., TF-IDF, mutual information) behave similarly. We decide for the TF-IDF weighting scheme for the terms. Moreover, the author limits the features of a single document to a maximum of 100 features. As a consequence, we extract from each document those 100 terms that have a maximum TF-IDF value and use the set of the terms of all documents as the entries in the document vectors. By running the k-means algorithm, the author strives for discovering the dominant topics in the document collection. (2) The second step is the extraction of cluster label candidates. The author experiments with frequent phrases or noun phrases as candidates. For our implementation of Descriptive k-means, we use the Apache OpenNLP library[3] to extract noun phrases as potential cluster labels. (3) The third step is the selection of those noun phrases that best match the terms and weights of the dominant topics' representations (the cluster centroid vectors). After representing both the label candidates and the cluster centroids using the Vector Space Model (VSM), the author determines those phrases that are most similar to the centroids. According to the original implementation, we utilize the Apache Lucene library[4] for retrieving the best matching candidate labels. Moreover, we apply the length penalty introduced by the author. That is, adjusting the retrieval score of each label candidate by penalizing it for being longer or shorter then the desired length of m terms. As already mentioned for the first baseline algorithm, we set m equal to the rounded average of the cluster labels of our approach, which is 3.18. The set of re-scored candidate labels is then sorted again and the highest scoring m phrases become the label for a given dominant topic. (4) In the last step, the author searches for cluster documents that are relevant to the selected labels in the previous step. Again, we utilize the Apache Lucene library and according to the author, we submit each label as a phrase query that matches documents containing all terms in the query, but allows reordering and injection of other words in between query's terms. The result list of the phrase query along with the query itself constitutes the labeled document clustering of this baseline algorithm.

[3] `http://opennlp.apache.org`, Last accessed: February 13th, 2015
[4] `http://lucene.apache.org`, Last accessed: February 13th, 2015

4 Evaluation

In this chapter, we evaluate our processing pipeline presented in Chapter 3. More precisely, we first introduce a novel data set that was especially generated for this thesis in Section 4.1. We then examine the three crucial steps in our pipeline and evaluate their various approaches by means of the novel data set. That is, the vocabulary generation methods are investigated in Section 4.2, while the different retrieval models are considered in Section 4.3. The evaluation of the constrained cluster analyses is described in Section 4.4. After determining the best setting for our approach, we compare it with two baseline algorithms in Section 4.5; among others, we conduct a user study to evaluate the explanatory power of a cluster label to its documents.

4.1 Ambient++ Data Set

The original Ambient data set was published by Carpineto and Romano in 2008.[1] Since then, it has become a popular data set for document clustering and labeling evaluation and has been utilized by a variety of research groups (e. g., [SGH11, SFMC12]). The Ambient data set comprises 44 ambiguous topics with about 18 subtopics each, which were obtained by exploring the topic's disambiguation page in Wikipedia. For instance, given the topic *Jaguar*, its disambiguation page provides subtopics for the cat, car manufacturer, game console, and others. Some of the subtopics are associated with a set of documents, which consists of an URL, title and snippet. The documents were collected by submitting every topic as a query to a web search engine in January 2008, and by assigning each URL of the top 100 result documents manually to its corresponding subtopic.

So far, the snippets are the only textual content provided by the data set. However, they are unequally distributed within the subtopics, and are also inappropriate to represent the actual document, because the clustering of short texts is difficult [PBR07]. As a consequence, we reconstruct the Ambient data set in such a way that the subtopics are equally assigned with documents of an appropriate length. We call the extended corpus the Ambient++ data set. The following six steps explain the evolution of our corpus.

[1] Claudio Carpineto, Giovanni Romano: Ambient Data set (2008), `http://credo.fub.it/ambient/`

Step 1: Acquisition

The related HTML documents of the original Ambient URLs form the basis of our corpus extension and are acquired in this step. The authors of the original data set assigned a total of 2,257 URLs to their corresponding subtopics. In doing so, some subtopics were provided with more URLs than others, which is due to the subtopics' different popularity and thus, the distribution of URLs in the result list of the requested search engine. In fact, the majority was equipped with no URL at all, while some were assigned with up to 76 URLs.

Step 2: Download

After acquiring the URLs, the next step is to download their related web pages. In this process, the following errors occur, resulting in only 1,697 downloaded documents.

- The document is not accessible because of redirection, client or server errors, as the date of acquisition dates back more than six years.
- The document has an invalid content type (neither *text/html* nor *application/x-html+xml*).

Step 3: Cleaning

Even if a web page is downloaded, their appropriateness for being a subtopic document cannot be guaranteed. The reasons for excluding a web page can be categorized as follows.

- Document-specific: These include server errors like the request from an unauthorized country as well as former pages of newspaper or search engines without mentioning the subtopic nowadays.
- Ambient-specific: The document is excluded because of a wrong language (i. e., not English), the assignment to a wrong subtopic and being either a disambiguation page that states multiple subtopics or a template page that is topic-dependent filled with numbers (e. g., weather pages of a town).

After a manual inspection of all web pages, 1,086 remain, while 611 were removed. The findings from Step 1 to 3 are illustrated in Figure 4.1. As the outliers indicate, the number of originally assigned documents per subtopic in Step 1 varies considerably, and the next two steps influence this distribution by decreasing the total amount of documents. Note that for the purpose of visualization, only subtopics with at least one assigned document are represented, which composes 44 % of all subtopics (349 of 790). The actual arithmetic mean for each step is shown as a filled dot. While there were 2.9 documents per subtopic originally assigned, this average is decreased to 1.4 after the cleaning step.

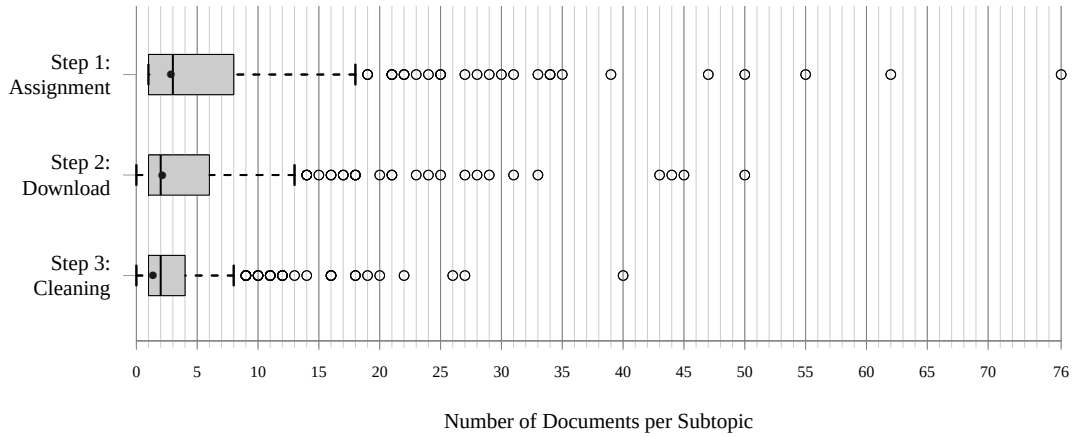

Figure 4.1: The distribution of documents per subtopic after the corresponding corpus extension step. The boxplots include only subtopics with more than one document, while the filled dots illustrate the overall arithmetic means.

Step 4: Enrichment

The goal of the enrichment step is to provide each subtopic with ten documents and thus to construct an equally distributed data set. The additional documents are obtained by submitting each subtopic description to a web search engine. While stepping through the corresponding result list, each document is then manually evaluated and assigned to its subtopic until ten documents are found. The restrictions of being a valid subtopic document are the same as those described in the first two steps; in addition, there are another two kinds of documents to exclude:

- The document consists exclusively of a list, table, image gallery or video and thus provides no textual content.
- The document was already assigned by the authors of the original data set.

In some cases, the provided subtopic descriptions fail to capture the actual meaning of the subtopic in a way that they can be used as a query. If the description is too specific and comprises multiple sentences, or the description is too general for not mentioning its actual topic, the search engine retrieves mostly invalid result documents. In both cases, mostly invalid result documents are retrieved from the search engine. For instance, requesting the subtopic

44.18 A comic book character owned by Marvel Comics

does not necessarily result in documents about the character and topic *Zombie*, since there are 1,478 different Marvel characters.[2] When recognizing an insufficient number

[2] `http://marvel.com/comics/characters`, Last accessed: February 13th, 2015

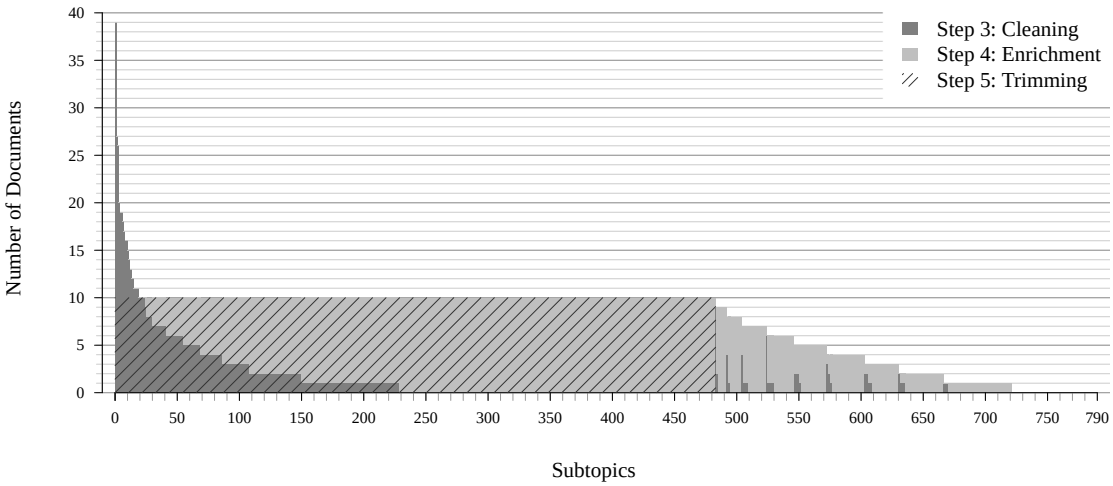

Figure 4.2: The two gray histograms show the number of documents for every subtopic after the corresponding corpus extension step. The hatched rectangle marks the trimmed range of documents.

of results that was due to a poor subtopic description, we manually formulate a more meaningful description and submit that to the web search engine instead. However, some subtopics are still too unpopular for retrieving ten valid documents.

In this step, another 4,506 documents were assigned to subtopics, leading to a total amount of 5,992 documents.

Step 5: Trimming

As stated above, not every subtopic can be sufficiently enriched. In contrast, there are still subtopics with more than ten documents, i. e., enough documents are retained during the download and cleaning phase. By applying the trimming step, a data set with subtopics of exactly ten documents is obtained. That is, discarding those subtopics with less and pruning those with more documents. For the latter task, the ten best ranked documents according to the search results of the original data set are retained.

Figure 4.2 illustrates the changing number of documents from the cleaning to the trimming step. While there were only 25 subtopics with a sufficient amount of ten documents after the cleaning step, these can be increased to 481 through the enrichment. The trimming step keeps this number since it only discards subtopics with less than ten documents. The kept subtopics of the trimming step are illustrated by the hatched rectangle. For this purpose, note that the subtopics are primarily sorted by the number of documents for the enrichment step, and secondarily for the cleaning step. Subtopics that are not within the rectangle, i. e., from 482 on, cannot be sufficiently enriched.

Step 6: Filtering

During the work in the enrichment step, several issues occurred. We therefore apply post-filtering mechanisms to counteract these issues and describe them in this section.

Even though the intermediate data set consists of subtopics with the same number of documents, there are sets of subtopics with obviously identical meanings; for example,

12.11 globe, a Japanese trance/pop-rock group and
12.17 globe (band), a Japanese pop music group.

That both subtopics refer to the same musicians is verified by investigating Wikipedia's revision history. When the subtopics were acquired from the disambiguation page for their topic *Globe*, they were linked to one and the same Wikipedia page. Moreover, some sets of subtopics form a series, making them inappropriate when the task is to differentiate between them. For instance, given the series

24.2 Metamorphosis I, a woodcut print by the Dutch artist [...],
24.3 Metamorphosis II, a woodcut print by the Dutch artist [...] and
24.4 Metamorphosis III, a woodcut print by the Dutch artist [...],

it is likely that documents about one subtopic also refer to the other. For the purpose of clustering these subtopics, only the root subtopic is retained (here, the first one). Note that the filtering of subtopics is only performed if they refer to one and the same topic. Since this data set is supposed to be clustered for each topic, thematically overlapping documents that are cross-topic do not influence the clustering; therefore, each subtopic is retained.

The filtering step reduces the total number of subtopics by 13, which leads to a total of 468 subtopics. Each topic covers at least three subtopics as illustrated in Figure 4.3. Here, every topic is represented by three stacked bars, each of it showing the number of subtopics after its corresponding step.

Content Extraction

So far, the Ambient++ data set comprises 4,680 HTML documents. In order to extract the textual content of the documents, we utilize the Boilerpipe library provided by Kohlschütter et al., which is based on detecting boilerplate code in HTML documents by using shallow text features [KFN10]. The library provides different approaches to extract either the plain text or the main content of the HTML documents. Since documents from different subtopics might be similar solely due to navigational or off-topic sections, we strive for extracting that text of a document which deals with its corresponding subtopic. After a manual inspection of the different main content extractors in Boilerpipe, we decide to choose the so-called Default Extractor and extract the main content for each of the 4,680 documents.

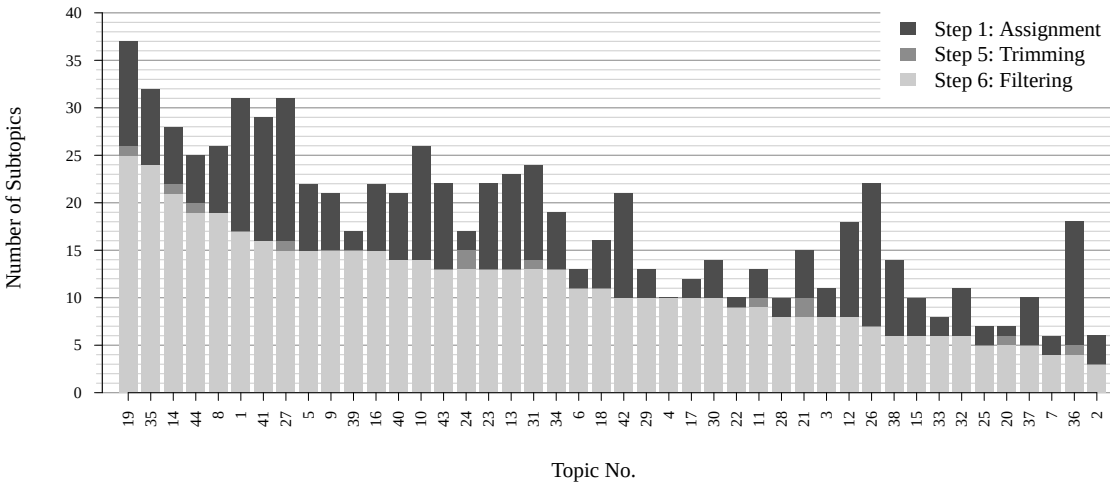

Figure 4.3: The number of subtopics for every topic after the corresponding corpus generation step. Other steps are not involved, since they do not influence this distribution.

4.2 Vocabulary Evaluation

The vocabulary of our approach is affected by two crucial factors: (1) the method of how to generate the entries in the vocabulary, i. e., whether they are automatically extracted from the documents or determined with regard to a predefined vocabulary and (2) the size of the vocabulary, i. e., the number of extracted phrases. The first factor is investigated in the next paragraph, whereas the latter is part of the paragraph that follows.

Generation Method

In Section 3.2, we introduced two different types of vocabulary generation methods: the identification of noun phrases using natural language processing, as well as the discovery of Wikipedia article titles in the documents. The purpose of the entries in the vocabulary are twofold: (1) They serve as search queries for the generation of postlists in the inverted index, i. e., the cluster documents and (2) they also constitute the cluster label candidates. With respect to the latter labeling task, the two different methods do not vary noticeably in their syntactical form since most Wikipedia article titles are noun phrases. It is more valuable to evaluate their feasibility to serve as queries that find similar documents. Therefore, we investigate the so-called Best Clustering that is possible when considering the inverted index. The next paragraph explains the process of finding the Best Clustering in more detail.

Best Clustering The Ambient++ data set consists of 44 topics and 468 subtopics with ten documents each. Each topic is considered as a single document collection, and the clustering task is to group the documents of a subtopic together in the same cluster. In this regard, the Best Clustering states the upper limit of all possible clusterings. It utilizes the documents of each subtopic as the gold standard (i. e., the truth information), and resorts to the inverted index to identify that query whose postlist best covers the truth documents; that is, whose postlist maximizes the F-measure (cf. Section 3.4.3) with respect to the original ten subtopic documents. Note that according to the query syntax constraint, the best query can also be a conjunction of two queries, and the corresponding postlist is the intersection of the postlists of both queries. The inverted index is highly dependent on the retrieval model of the search engine. Since we want the retrieval model to have the smallest possible influence on the evaluation of the vocabulary, we decide to utilize the most simple retrieval model, i. e., the BOOLEAN model (cf. Section 3.3). Moreover, we do not apply a relevance constraint, so that every document retrieved by the search engine is stored in the inverted index. To summarize, the Best Clustering consists of one (conjuncted) query per subtopic as cluster labels, and the relevant postlists constitute the clusters' documents.

While the Best Clustering represents a small subset of the most suitable queries in the inverted index, another approach is to consider the inverted index as a whole. That is, each entry in the index serves as a cluster. We refer to this approach as the Inverted Index Clustering, and describe the rationale and the evaluation process in the next paragraph.

Inverted Index Clustering The inverted index lists all vocabulary terms along with their retrieved documents, including very specific terms with only few documents, as well as very broad terms that cover more than one subtopic and are thus likely to retrieve more than ten documents. When considering the inverted index as a whole, each entry serves as a cluster. Rationale for this approach is the assumption that the entries of the inverted index can be seen as the support of the index for the cluster analysis. The more queries retrieving the same documents, the more likely these documents are grouped together in one cluster (e. g., see the positive rate for the set cover clustering in Section 3.4.1). Assuming that a lot of entries retrieve only documents from one subtopic, the support of this particular vocabulary is high since the cluster analysis can easily identify these documents as being very similar. On the contrary, a vocabulary with a lot of entries retrieving documents from multiple subtopics has a low support, since the cluster analysis is impeded from identifying similar documents (i. e., documents from the same subtopic). In contrast to the Best Clustering evaluation, it is unknown to which subtopic a query and its documents belong, i. e., the truth information is missing. For example, given a query "animal" that retrieves documents from both

Table 4.1: Example clustering for calculating the Soft F-measure, where a value of 1 indicates the membership of the document to a cluster.

Cluster	Documents			
	a_1	a_2	b_1	b_2
A	1	1	0	0
B	0	1	1	1

subtopics “Camel, species of humped ungulates” and “Camel spider, an order of arachnids”, this query cannot be assigned to either the first or the second subtopic; therefore, no truth information can be obtained and no conventional F-measure can be calculated. We hence need a measure that is capable of evaluating such ambiguous and overlapping clusters.

Soft F-measure In order to demonstrate how our novel measure evaluates an overlapping cluster, Table 4.1 lists an example clustering consisting of two clusters. While cluster A contains two documents, namely a_1 and a_2, cluster B contains three documents, namely a_2, b_1 and b_2. The cluster and document names already reveal their corresponding subtopic. That is, cluster A exclusively contains correct documents, while cluster B is impure for containing one incorrectly assigned document (a_2). Picking up the aforementioned example query and subtopics, cluster B could have been retrieved by the query ”animal“, leading to documents about the ungulate as well as the arachnid. Our novel measure is based on the conventional F-measure, for it is also aimed at detecting true or false positives (tp and fp), and true or false negatives (tn and fn). A major difference is that our measure evaluates pairs of documents in a clustering, while the conventional F-measure evaluates the correct assignment of a document to a cluster (with the help of a truth information). Moreover, it is no longer a binary decision between either true or false but rather a continuous degree, depending on the ratio of shared clusters. For each document pair in the clustering, we calculate their association strength by the fraction of shared clusters divided by the union of their clusters. In other words, when considering the document columns in Table 4.1 as bit vectors, the association strength of two documents is the AND operation divided by the OR operation of their bit vectors. After calculating the association strength, we resort to the truth information of the data set and reveal if the two documents are in the same subtopic or not, i. e., whether their association strength is justified or not. In the former case, the value of the association strength is added as a true positive, and in the latter case, as a false positive. Note that the maximum association strength equals 1, that is, the documents are exclusively in the same clusters. Therefore, the value of the false negatives for this document pair equals 1 minus the true positives, and the value for the true negatives

Table 4.2: Example confusion matrix for calculating the Soft F-measure. The gray formulas denote the summands of the relevant document pairs.

		Truth	
		True	False
Clustering	True	$tp = \frac{1}{2} + 1 = 1.5$	$fp = 0 + 0 + \frac{1}{2} + \frac{1}{2} = 1$
	False	$fn = \frac{1}{2} + 0 = 0.5$	$tn = 1 + 1 + \frac{1}{2} + \frac{1}{2} = 3$
		$a_1a_2 + b_1b_2$	$a_1b_1 + a_1b_2 + a_2b_1 + a_2b_2$

equals 1 minus the false positives. We call the novel measure "Soft F-measure" with regard to the term "soft clustering algorithms", which denotes a group of algorithms that find clusterings where a document has a fractional membership in several clusters.

To illustrate this calculation, Table 4.2 shows the confusion matrix (a compact 2x2 table for listing tp, fp, fn and tn) of the example clustering. The gray formulas under the table denote the summands of the relevant document pairs. The documents a_1 and a_2 share one cluster (A), and document a_2 is additionally part of another cluster (B). Since both documents belong to the same subtopic, their association strength of $\frac{1}{2}$ is a true positive, and the remaining $\frac{1}{2}$ constitute a false negative. The portion of every document pair is then added up, and the F-measure of this confusion matrix is:

$$F_1 = 2 \cdot \frac{0.6 \cdot 0.75}{(1 \cdot 0.6) + 0.75} = 0.\overline{6}.$$

The above example denotes only one calculation of the Soft F-measure. In order to obtain a more reasonable insight, Figure 4.4 shows the graph of a cluster for a subtopic whose documents change over time. Of the originally ten documents, the cluster only includes one at the beginning ($x = 1$). Then, the cluster size increases stepwise by one document until all ten documents are included ($x = 10$). Afterwards, the cluster becomes more and more impure by adding documents that do not belong to the subtopic. At the end, the cluster contains ten correctly and ten incorrectly assigned documents ($x = 20$). As the graph for the cluster reveals, the Soft F-measure is not symmetric, so only retrieving six of ten documents performs worse than retrieving all relevant documents and four irrelevant documents.

While we are able to evaluate the Best Clustering with the conventional F-measure due to the truth information for a document-cluster pair, the Soft F-measure allows us to evaluate the inverted index by evaluating document-document pairs. Table 4.3 lists the evaluation of both vocabulary generation methods with respect to the Best Clustering and the Inverted Index Clustering, respectively. More precisely, the first row denotes

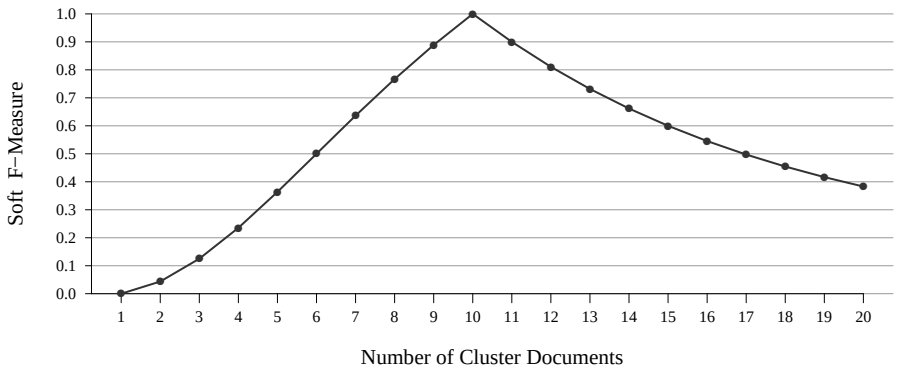

Figure 4.4: Example graph for justifying the Soft F-measure. After ten relevant documents are stepwise assigned to a cluster ($x = 10$), this cluster becomes more and more impure by adding stepwise irrelevant documents.

the average value for the (Soft) F-measures of all 44 topics (AVG), whereas the second row denotes the corresponding standard deviation (SD). As the measures reveal, both vocabularies contain entries that are capable of retrieving almost all documents of each subtopic. Nevertheless, noun phrases seem to constitute better queries, as their average F-measure is slightly higher while having also a lower standard deviation. The same holds for the average Soft F-measure and the inverted index, even though the differences are even smaller. As expected, the average Soft F-measure for the inverted index is noticeably lower. This is based on the fact that the index also contains very general entries that often not only retrieve documents from one subtopic, or very specific entries that only retrieve a subset of the documents from a subtopic. The Wikipedia article titles not only achieve slightly poorer results but they also require higher computational effort to be generated. We therefore decide to utilize noun phrases as the vocabulary to serve as search queries as well as cluster labels.

Table 4.3: Comparison of the average F-measures (AVG) for noun phrases and Wikipedia article titles with regard to the Best Clustering as well as the Inverted Index Clustering. In addition, the standard deviation is listed as SD.

	Best Clustering		Inverted Index Clustering	
	Noun Phrases	Wikipedia Articles	Noun Phrases	Wikipedia Articles
AVG	0.9293	0.9128	0.2553	0.2452
SD	0.0440	0.0487	0.0604	0.04939

Extracted Terms

The objective of the evaluation in this section is to obtain a suitable number of phrases to be extracted from each document. When reducing the number of extracted phrases, the number of queries submitted to the search engine are also reduced and thus, the overall performance of our approach is increased. Rationale for the limitation is to keep valuable phrases, i. e., those that describe the document and hence the subtopic best. For each vocabulary generation method, Section 3.2 introduced a measure to rank the phrases according to their relevance. We extract a range of 1 to 20 phrases from all documents and evaluate the Best Clustering as well as the Inverted Index Clustering as described in the previous section. Figure 4.5 shows the resulting graphs of both evaluations. Since the results for the noun phrases as well as for the Wikipedia article titles are comparable, we only depict the results for the noun phrases. The F-measures for the Best Clustering and the inverted index are plotted as solid lines with regard to the left y-axis, whereas the right y-axis belongs to the size of the vocabulary shown as a dashed line. As the figure reveals, the F-measure of the Best Clustering saturates at an extracted number of about six noun phrases, although the vocabulary size is continuously increasing. This confirms our intent to limit the extracted phrases, since additional extracted phrases expand the vocabulary unnecessarily, without producing further valuable search queries or cluster labels, respectively. This observation is also supported by the graph of the Inverted Index Clustering. Since it is continuously decreasing, the majority of further extracted phrases does retrieve either less subtopic documents or documents from multiple subtopics. However, several phrases cover their subtopic better than the previously extracted phrases, since the F-measure of the Best Clustering increases. To summarize, we decide to extract six phrases from each document in the collection, which denotes a suitable ratio of valuable phrases to the size of the vocabulary.

4.3 User Model Evaluation

After extracting six noun phrases from each document, the resulting vocabulary is then submitted to a search engine in order to retrieve relevant documents for each phrase. Although the extracted phrases denote the most valuable phrases for the corresponding document, it is not guaranteed that they are exclusively about the document's subtopic. In case that the document is very short, several of the six extracted phrases might be very general, as the document does not provide enough subtopic-specific phrases. On the contrary, very long documents might lead to extracted noun phrases that are only document-specific and do not cover the relevant subtopic. In other words, it is challenging to evaluate the user model with insufficient phrases at hand, for we are not able to ensure the phrase's relevance to the subtopic. In order to overcome this impediment, we manually generate appropriate phrases for each of the 468 subtopics in the Ambient++ data set. More precisely, we iterate through all 44 topics and their

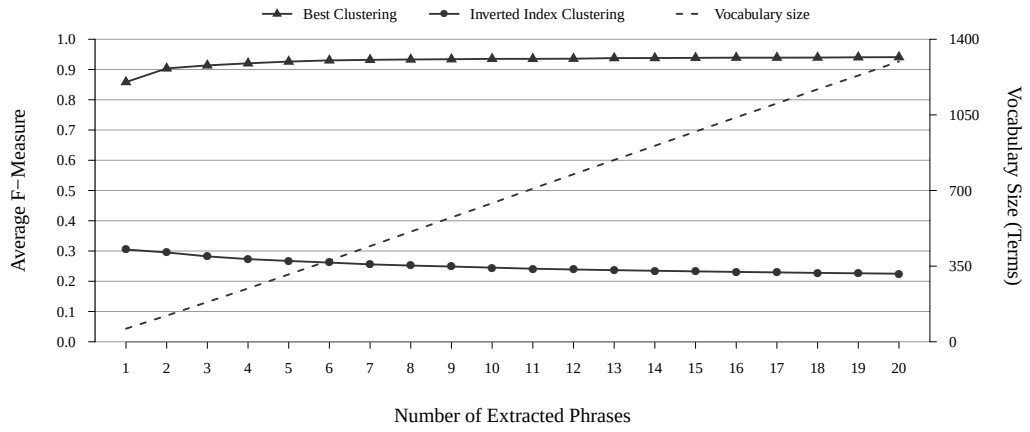

Figure 4.5: F-measures for the Best Clustering as well as the inverted index for a different number of extracted phrases (left y-axis) along with the vocabulary size (right y-axis).

corresponding subtopics, and create a phrase of one to five words that is based on the subtopic's description from the original Ambient data set. For example, given the subtopic

16.2 Jaguar (car), a British luxury car manufacturer, owned by Ford as of 1990,

the generated phrase is "car manufacturer". Note that we aim at describing the subtopic in order to distinguish it from the other subtopics of the topic and do not include the topic itself (here, "Jaguar").

In the remainder of this section, we first evaluate the different retrieval models for the search engine. We then examine the two relevance constraints to trim the result list and finally, we investigate the different concept extraction methods for the ESA model.

Retrieval Model

The retrieval models presented in Section 3.3 strive for modeling the user in the decision of whether a document is relevant to a given query or not. For their evaluation, the BOOLEAN model constitutes the baseline algorithm, and the objective of the three retrieval models, TF-IDF, BM25 and ESA, is to outperform its retrieval quality. We therefore submit each manually generated query to four search engines, each of them utilizing one of the retrieval models. We then examine the F-measure of the result list in respect of the subtopic the query belongs to. With regard to the relevance constraint, we trim each result list to the best possible rank to cut the result list, i. e., to that document which maximizes the F-measure. Thereby, we obtain the best possible performance of the retrieval model without being influenced by one of the introduced relevance constraints. Note that the last section of the user model evaluation investigates to what extent one

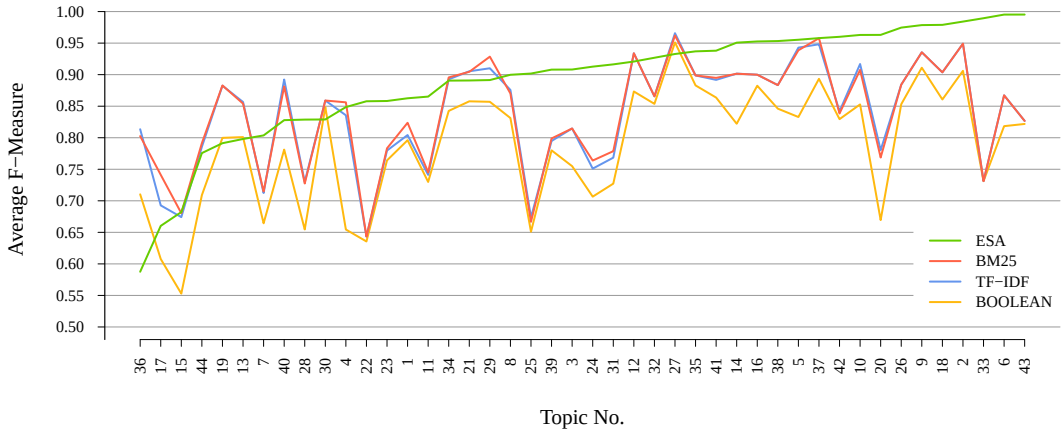

Figure 4.6: Performance of different retrieval models. Each topic is represented by the average F-measure of the result list of the subtopics. The result lists were retrieved by submitting the corresponding manual query.

of these constraints reaches this best rank. Figure 4.6 shows the average F-measure of each retrieval model for every topic. Irrespective of a few outliers, all three retrieval models outperform the baseline BOOLEAN model. Note that the graph characteristics for the three bag-of-words models, BOOLEAN, TF-IDF and BM25, are comparable. The only difference in their performance is based on their different ranking of the documents. While the result list of the BOOLEAN model cannot be ranked and therefore not suitably trimmed, the different ranking functions of the latter models do not influence the retrieved and trimmed result list to a great extent. Besides the possibility to rank and trim the result list, TF-IDF and BM25 allow a partial matching of the query terms, leading to documents in the result list that are not retrieved by the BOOLEAN model. As the best retrieval model we determined the ESA model, for its results outperform the performance of the other models in 35 of 44 topics. Note that the topics in the figure are sorted by the performance of the ESA model. Although a similar ascending trend of the F-measure for the other models is noticeable, the performance differences are subject to considerable fluctuations. This is based on the fundamental differences in the retrieval process. While bag-of-words models directly examine the query with regard to a document, topic models like ESA compare their similarity with regard to high-level concepts, independent of overlapping terms between the query and the document. As the figure reveals, such an approach is eligible most of the times, but also fails sometimes to establish a border between certain subtopics. For example, consider topic 36, “The little mermaid”. Its subtopics include a statue in Copenhagen, an animated film from Walt Disney Pictures, and a Broadway stage musical. They are all based on the novel by the Danish author Hans Christian Andersen, which is the last of the overall four subtopics. The manual

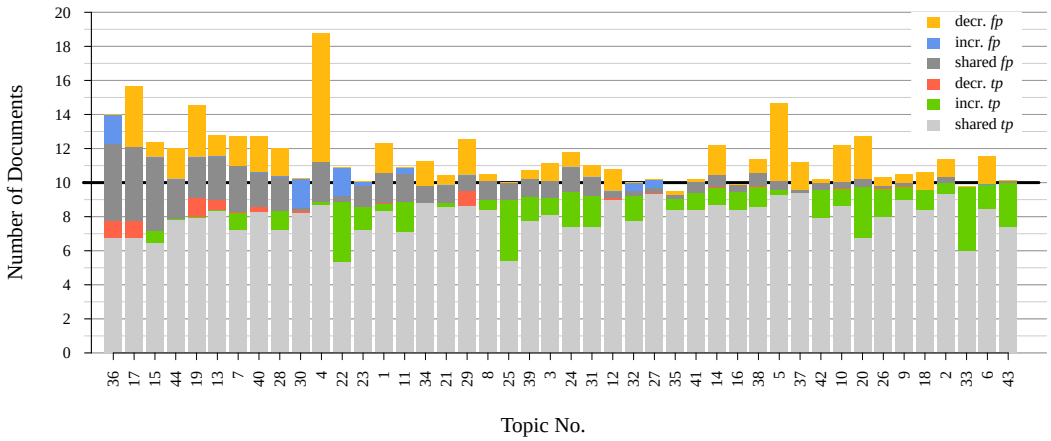

Figure 4.7: Changes in the result list from BOOLEAN to ESA model. Each topic is represented as a stacked bar of shared true positives (tp) and false positives (fp). Colored bars indicate the decreasing or increasing changes from ESA to the BOOLEAN model.

query for the latter subtopic is "novel by hans christian andersen", which will activate concepts that are about novels or mention Hans Christian Andersen. The crucial point is that also documents of the other subtopics are likely to refer to these concepts due to their adoptions from the original novel. Depending on the amount of references to the discovered concepts, the similarity between an irrelevant document and the query increases and might be part of the trimmed result list. On the one hand, the user can benefit from retrieval models that discover hidden relations between queries and documents. On the other hand, in scenarios like ours, we strive for separating the different subtopics from each other as strict as possible, and the strength of topic models can be hindering.

However, in most cases the retrieval performance can be significantly increased. The difference is especially noticeable when comparing ESA with the baseline BOOLEAN model. Therefore, Figure 4.7 illustrates the improved performance and shows the changes in the result list for each topic. Again, the topics are sorted by the F-measure of ESA in ascending order. Each topic is represented as a stacked bar of true positives (retrieved documents that belong to the manual subtopic query, shown in light gray) and false positives (retrieved documents that do not belong to the manual subtopic query, shown in dark gray). In addition, colored bars indicate the changes from ESA to the BOOLEAN model. While the red and green bars illustrate a decrease or increase of true positives, the orange and blue bars show a decrease or increase of false positives. Note that the retrieval performance benefits from both an increase of true positives and a decrease of false positives. In the following, we spotlight two example topics: (1) Topic 36 illustrates the worst case of a performance loss (the leftmost topic). Not only that for each subtopic

two more irrelevant documents are retrieved on average (blue bar), but ESA also retrieves one relevant document less (red bar). (2) An opposed example is given by topic 6 (second from the right). With ESA, the false positives can be reduced by 1.5 documents on average, whereas the true positives can be increased to almost ten documents. Note that each subtopic consists of ten documents; hence, ESA retrieves for each subtopic of topic 6 almost every relevant document without any irrelevant document.

To conclude, we utilize the ESA model to retrieve relevant documents for each phrase of the vocabulary.

Concept Extraction for ESA

The retrieval of documents with the help of ESA requires high-level concepts to which the query and the documents can be compared. As introduced in Section 3.3.3, we utilize Wikipedia articles as concepts. However, articles in Wikipedia are known to be of arbitrary length and therefore address the topic of the articles to different extents. While longer articles also cover the wider scope of the topic and make references to rather general subjects, short articles tend to be very specific and only consider the directly related subjects, if any. Therefore, using long articles as concepts might lead to similarities to a number of documents in the collection, while short articles might be activated from only few documents. The objective of the evaluation in this section is to investigate the two extraction methods for Wikipedia articles presented in Section 3.3.3; that is, the mere extraction of the full content of an article versus the deliberate extraction of the highly relevant lead section and extending it with the first sentence of linked articles. For the evaluation process, we submit the manual query for each subtopic (cf. Section 4.3) to a search engine using the ESA model and either the full articles or the extended lead section as the concept space. We then average the F-measures of the subtopics for each topic. The outcome of the investigation is given in Figure 4.8. The topics are sorted by the average F-measure of the full article extraction in ascending order. Note that this extraction method was already utilized in the previous section, which is the reason for a similar graph in Figure 4.6 and Figure 4.8. The graph at hand shows that the use of the extended lead sections as concepts does not increase the retrieval performance, but decreases it. A more detailed investigation reveals that a loss of true positives (i. e., relevant documents) is the major reason for the decreased F-measures, while the false positives (i. e., irrelevant documents) are only slightly increasing. This leads to the assumption that the shortened content causes a loss of valuable connections between the concept and the relevant documents. In other words, the lead section does not provide enough valuable content to distinguish the documents of the different subtopics; at least not to that extent as the full article distinguishes the documents.

To summarize, we extract the full content of each Wikipedia article, which will then serve as a concept for ESA. Since we know that all documents deal with one of the subtopics of the Ambient++ data set, we can incorporate this domain knowledge by

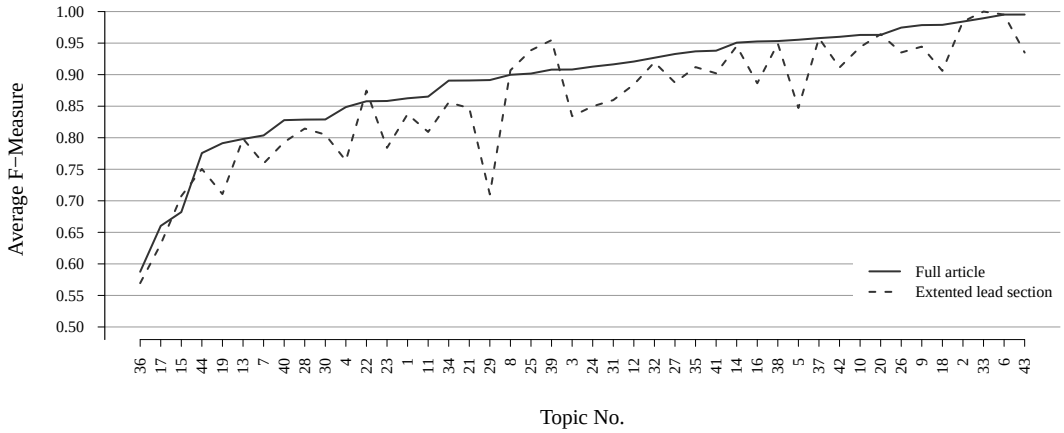

Figure 4.8: Comparison of different content extraction methods for the Wikipedia articles, which serve as ESA concepts. For each topic, the F-measure of its subtopics is averaged.

choosing the Wikipedia articles for ESA accordingly. More precisely, we obtain the corresponding Wikipedia article for each subtopic from the English Wikipedia dump from December 8, 2014. If there is no Wikipedia article for a subtopic, we merge randomly selected passages of the ten subtopic documents, and use this collection as the corresponding concept. To this point, the concept space comprises 468 concepts. In their work, Anderka and Stein suggest to use 1,000 to 10,000 concepts, as this denotes a reasonable trade-off between accuracy and runtime [AS09]. We therefore extend the concept space with further, randomly selected Wikipedia articles, leading to a total amount of 5,000 concepts.

Relevance Constraint

One of the major advantages of the TF-IDF, BM25 and ESA model over the BOOLEAN model is the possibility to rank the result list retrieved by the search engine. With the ranking of documents according to their relevance score, we are able to trim the result list and therefore ensure that a query only retrieves those documents which are *about* the query topic, i. e., the documents which are returned in the top results of a query. However, determining the position in a query result list where the mere occurrence of the query phrases turns into "aboutness" is not trivial. Section 3.3.1 introduced two relevance constraints that address this problem. While the top-k constraint trims the result list to a given rank k, the score constraint trims the result list to a variable rank k that is determined by calculating a minimum relevance score that has to be exceeded by the retrieved documents.

The evaluation process of both constraints is as follows. For each of the 468 topics, we first submit the manual generated query (cf. Section 4.3) to the search engine and

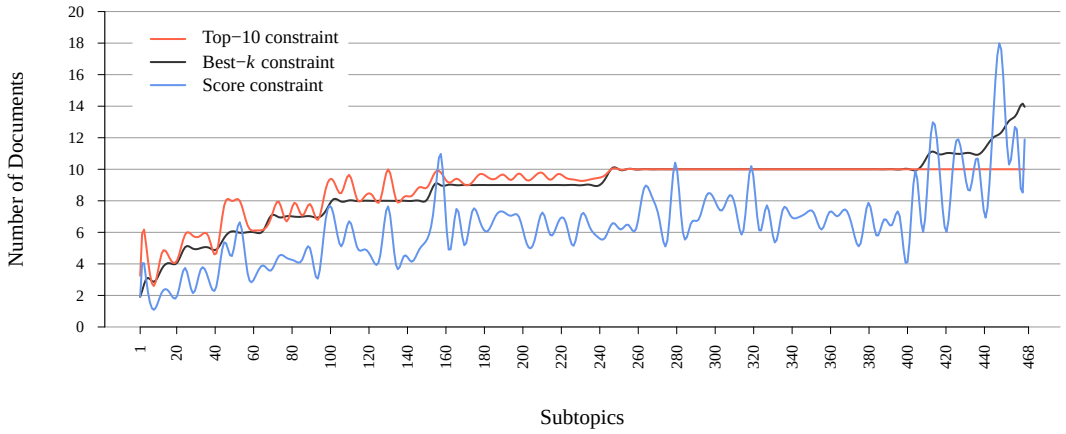

Figure 4.9: Length deviation of the results lists for different relevance constraints to the best-k constraint using the BM25 model.

determine the rank at which the constraints cut the result list. Note that the top-k constraint cannot always trim the result list to rank k, but only to the minimum of the result list size and k. As described in Section 3.3.1, the top-k constraint limits the number of documents in a cluster to k. Since we know that each subtopic consists of ten documents, we set $k = 10$. The next step is to calculate the best possible rank to cut the result list, i. e., to that document which maximizes the F-measure with regard to the subtopic of the manual query. We call this trimming method the best-k constraint. Figure 4.9 shows the cut-off ranks of each trimming method when submitting the 468 manual queries to a search engine utilizing the BM25 model. The subtopics are sorted by the best-k rank in ascending order. Moreover, we slightly smooth the resulting graphs for the sake of a better visualization; however, the expressiveness of the figure is not affected. The outstanding characteristic of the graph is that the score constraint nearly always fails to cut the result list at the best rank, whereas the top-10 constraint only slightly deviates from the best rank. Nevertheless, the negative impact of strictly trimming the list at a certain rank is unveiled when considering the last 60 subtopics. While the best-k increases to ranks higher than ten, the top-10 constraint still cuts the list at ten documents and cannot tap the full potential. As the graphs already indicate, also the numerical differences are noticeable. The top-10 constraint deviates from the best rank by 0.51 ranks on average (top-11 deviates 0.65 ranks, top-9 deviates 0.85 ranks), whereas the score constraint deviates by 2.96 ranks. In order to examine the poor performance of the score constraint, we determine the size of the unconstrained result list, which turns out to be 10.64 documents on average. The score constraint inevitably has to cut the list at a rank lower than 10.64 due to its (also modified) averaging calculation of the retrieval scores. Obviously, the calculated score threshold is nearly always not high

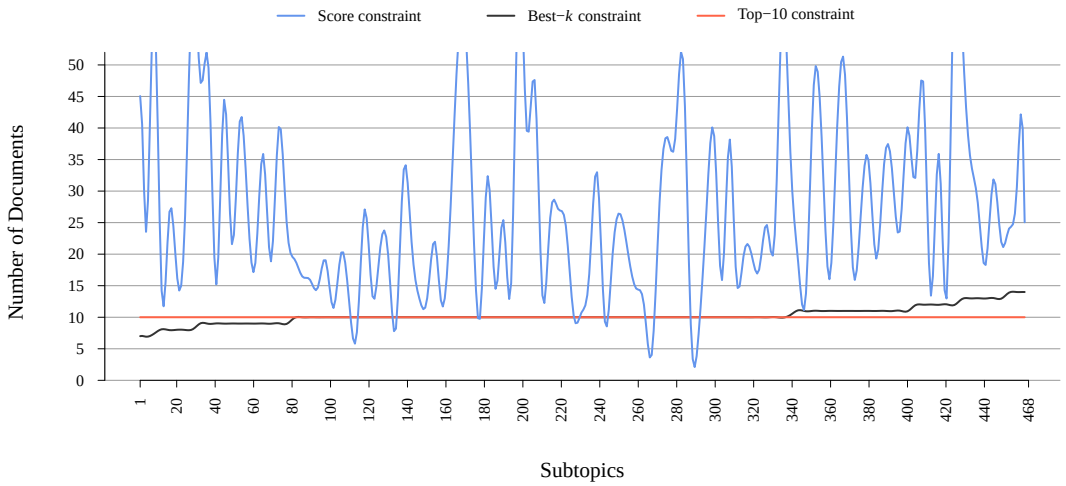

Figure 4.10: Length deviation of the results lists for the top-10 and score constraints to the best-k constraint using the ESA model.

enough and too many relevant documents are discarded for reaching the average best rank at 8.82. A more detailed view on the retrieval scores of the unconstrained result list reveals that the average relevance scores is 6.46 with a rather small standard deviation of 1.71. That is, almost every retrieved document has a similar relevance for the query. With such a result list, the score constraint calculates a threshold that easily gets below the score of the best rank, which 7.06 on average. Further investigations on the influence of the search engine and the quite similar retrieval scores of both relevant and irrelevant documents go beyond the scope of this thesis. However, we are confident that a more distinct deviation of the retrieval scores would also lead to a better performance of the score constraint.

The demand for a more distinct separation of the retrieved documents is all the more greater for the ESA model. As Figure 4.10 reveals, the score constraint for ESA performs even worse than for the BM25 model. The major reason for the poor performance is based on the long unconstrained result list (74.94 documents on average). Note that it was a known problem that ESA discovers at least a very small similarity to almost every document and which was one of the reasons to modify the simple average calculation. However, the rather small standard deviation of 0.14 for such a long result list at an average retrieval score (i. e., cosine similarity) of 0.32 leads this time to a too high score threshold, resulting in too many documents that are still retrieved irrespective of the score constraint. The rank deviation to the best-k constraint is 17.62. In contrast, the top-10 constraint performs almost as good as for BM25 with a rank deviation to the best-k constraint of 0.84.

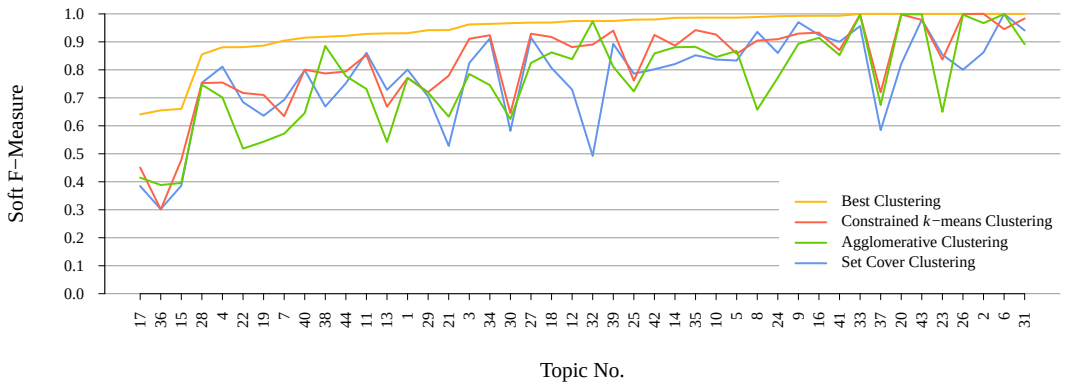

Figure 4.11: Evaluation of our different clustering algorithms. In addition, the Best Clustering is shown, which denotes the upper limit of the evaluation.

To conclude, the top-k constraint with an appropriate value for k performs better than the score constraint. However, the former cannot lead to a cluster size greater than k, which can be an undesired scenario in certain applications where some clusters should contain more documents than k and some not. We therefore introduce a first approach to obtain a variable rank k, though it cannot reach the performance of the top-k constraint for our data set. For the remaining evaluations in this chapter, we choose the top-10 constraint to cut the result list and obtain only highly relevant documents in the postlists of the inverted index.

4.4 Clustering Evaluation

In this section, we evaluate the three clustering algorithms introduced in Section 3.4. They all satisfy the common-query constraint in order to group similar documents in the same cluster. That is, only documents that are retrieved for the same query are candidates to be merged into the same cluster.

The evaluation process is as follows. After each algorithm has found its clustering for every topic, we use the Soft F-measure (cf. Section 4.2) to evaluate the correct grouping of the subtopic's documents in the same clusters. More precisely, we average the Soft F-measures of the clusters that were found for a topic. In addition, we generate the Best Clustering for each topic that is possible when trying to cover the truth documents with the document postlists in the inverted index (again, cf. Section 4.2). For the ability to cross-compare the evaluation of the Best Clustering and our algorithms, we do not evaluate the Best Clustering with the conventional F-measure as done in the vocabulary evaluation in Section 4.2, but with the Soft F-measure as well.

Figure 4.11 depicts the resulting Soft F-measures of each clustering algorithm. The topics are sorted by the performance of the Best Clustering in ascending order. The

graph for the Best Clustering reveals that the perfect clustering (i. e., every subtopic document is exclusively grouped with the other subtopic documents in one cluster) can only be found for eleven clusters (Soft F-measure equals 1.0). For the remaining topics, the inverted index does not provide a "perfect" query for each of the corresponding subtopics. Our three algorithms fail to find the est possible clustering for most of the topics, but still perform well. For example, the Constrained k-means clustering has an average Soft F-measure of 0.83, which denotes a deviation of 0.11 percentage points from the Best Clustering. With reference to Figure 4.4 in Section 4.4, such a value for the Soft F-measure states a result list that has a ratio of subtopic to non-subtopic documents of 9:0 or 10:1. The agglomerative clustering and the set cover clustering also perform with an acceptable Soft F-measure of 0.77 and 0.76, which both denote a ratio of 8:0. However, for its smallest deviation from the Best Clustering, we choose the Constrained k-means clustering to find clusters of similar documents.

In order to summarize the evaluations for each of the pipeline steps, we list the corresponding findings and give thereby an overview of our approach.

Step 1 We extract the best six noun phrases of each document in the collection, where each phrase has a maximum length of four terms. The set of phrases serves as the vocabulary for the cluster labels.

Step 2 We submit each noun phrase in the vocabulary as a query to a search engine that utilizes the ESA model. The concept space is obtained by extracting the full content of 5,000 Wikipedia articles. The resulting inverted index is expanded with the conjunction of each phrase pair along with the intersecting documents in their result list.

Step 3 We trim each result list in the inverted index to the top-10 documents and thereby keep only highly relevant documents for each query.

Step 4 We find clusters of similar documents by using the Constrained k-means algorithm, which utilizes the keyqueries of a document as its features. The found queries for the clusters serve as the cluster labels, while their corresponding postlists serve as the cluster documents. Therefore, the common-query constraint is satisfied, which demands the presence of a shared query between the cluster documents.

These four steps form the pipeline of our approach, which is evaluated with regard to the baseline algorithms in the next section.

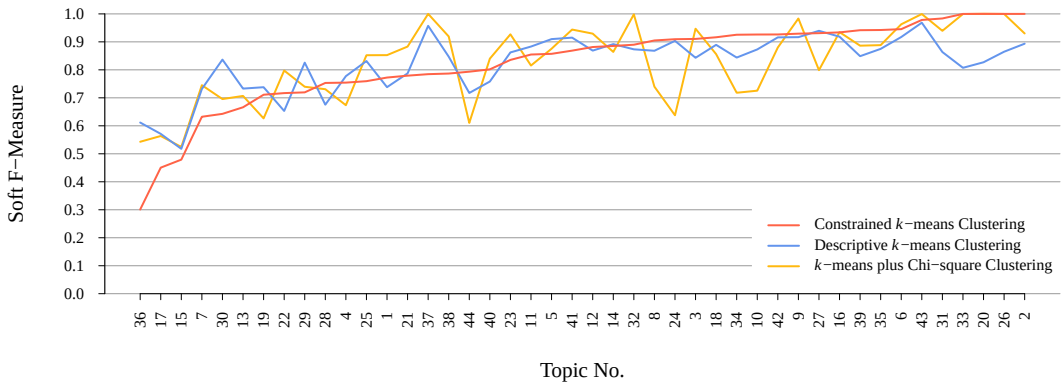

Figure 4.12: Comparison of our clustering algorithm (red line) with the two baseline algorithms.

4.5 Baseline Evaluation

The characteristics of a good document clustering algorithm are twofold: (1) The cluster documents have a high similarity to each other and a low similarity to documents in other clusters and (2) the cluster label describes each document in a cluster and distinguishes them from those in other clusters. In this section, we compare our approach to two baseline algorithms described in Section 3.5. While the first part of this section deals with the evaluation of the cluster documents, the last part of this section describes the evaluation of the cluster labels.

Cluster Documents Evaluation

The evaluation process of whether the documents of a subtopic are grouped together in the same cluster is identical with the evaluation of our different clustering algorithms in the previous section. That is, we evaluate the clustering of each algorithm by calculating the Soft F-measure for the clusters that were found for a topic. Figure 4.12 illustrates the findings of the evaluation. The topics are sorted by the performance of our Constrained k-means clustering in ascending order. Although the figure reveals quite different Soft F-measures for several topics, the algorithms perform almost identical with regard to the average Soft F-measure for all topics. The Constrained k-means as well as the k-means plus chi-square test algorithm perform with an average Soft F-measure of 0.83, whereas the Descriptive k-means algorithm deviates only marginally with an average Soft F-measure of 0.82. Note that we set k to the number of subtopics for all three algorithms.

Despite the deviations for the subtopics, a noticeable characteristic is that an increase of the Soft F-measure of our approach generally indicates a performance increase of the other algorithms as well. That leads to the assumption that the Ambient++ data set

provides several topics that are generally challenging to be clustered into their subtopics. Note that it is not an issue of the used k-means algorithm for all approaches, since the features for the analysis are entirely different, and our other algorithms (which are not based on k-means) reveal the same characteristics (cf. Figure 4.11). In order to support the assumption of challenging topics in the Ambient++ data set, we investigate the influence of the number of subtopics per topic, i. e., the number of clusters the algorithms shall find. Therefore, we sort the topics according to their average Soft F-measure of all three algorithms, and calculate the average number of subtopics of those topics that (1) are below the average of all topics and (2) higher than the average of all topics. The evaluation reveals that the number of subtopics per topic has no influence on the overall ability to cluster the topics, since the averages are 10.33 and 10.85, respectively, and an almost equal standard deviation of 5.0 and 5.1, respectively. Moreover, we examine to what extent the topics below the average consist of such challenging subtopics as it is topic 36, "The little mermaid" (a movie, musical and statue based on the novel), which was already subject of the retrieval model evaluation in Section 4.3 and which has the smallest average of all topics (the leftmost topic in Figure 4.12). The examination shows that there is only topic 15 ("Iwo Jima"), which has similar related subtopics (several movies, a ship and a photograph that are all about the battle fought on the island Iwo Jima). Although this topic is below the average, there are many other poor performing topics that have a clear separation of their subtopics. However, content-related subtopics are surely an impediment for finding a good clustering.

To summarize, our approach performs well enough to compete with the baseline algorithms and find clusters of comparable high quality. However, the cluster documents are only one of the factors that constitute a good document clustering that aims at supporting the users for navigational tasks. An even more important factor are the cluster labels—after all, the users first read the labels before choosing the desired cluster. As a consequence, the next section evaluates the corresponding labels of the clusters generated in this section.

Cluster Label Evaluation

The appropriateness of a cluster label to the cluster documents is a challenging characteristic to evaluate, since there is no truth information in the data set of whether a label covers the documents or not. Moreover, the users that are confronted with the labels might have different expectations for the labels depending on their own vocabulary and knowledge base. To empirically access the appropriateness of the cluster labels generated by the approaches, we conduct a user study with two subsequent experiments that evaluate (1) the discriminative power and (2) the descriptive power of the cluster labels. We again use the Ambient++ data set for the experiments.

Experiment 1: Discriminative Power

In the first experiment, we examine to what extent the cluster labels can discriminate the documents from their cluster to the documents from other clusters. Our hypothesis is that using our approach increases the discriminative power of the cluster labels when compared to the baseline algorithms.

Experimental Design To test our hypothesis, we conduct an empirical browser-based study in a within-subjects design [LFH10]. This means that our participants are exposed to every treatment, i. e., to every approach to be evaluated. Rationale for the experiment is that a participant is confronted with the cluster labels of a clustering of one algorithm, and has to select that label which best fits a given cluster (forced-choice). The cluster is represented by the subtopic that it is supposed to cover. That is, we iterate through all subtopics of the topic and select for each cluster that subtopic which maximizes the F-measure with regard to the cluster documents. For reasons concerning the duration of the evaluation, we only consider a subset of the topic clusterings for the evaluation and randomly select 22 of the 44 topics in the Ambient++ data set. In addition, we limit the given clusters that have to be discriminated from the others to four. More precisely, we determine those four subtopics that have the maximum average F-measure for all approaches. Thereby, we strive for ensuring a comparable high performance of all approaches. In fact, every approach has an F-measure higher 0.8 for each of the selected subtopics. However, we not only present the selected labels to the participants, but also the remaining cluster labels of the clustering, so that the participants have to decide between a maximum of eight labels (some clusterings have less than eight clusters). To this step, our evaluation corpus comprises 22 topics à four subtopics (some topics have only three subtopics, so that the average of subtopics per topic is actually only 3.77). For each subtopic three cluster labels exist—one for each of the three approaches. In order to obtain a reliable judgment for every subtopic, each subtopic is evaluated by three different participants. In total, we have 249 judgments for every approach (22 topics $\cdot$ 3.77 subtopics $\cdot$ 3 judgments).

In order to present the subtopics to the participants, we do not utilize the subtopic descriptions of the original Ambient data set, since they often fail to briefly cover the subtopic in only few words ($AVG = 9.1$ words, $SD = 4.4$ words, $MAX = 49$ words). Instead, we manually generate short descriptions for the subtopics. We therefore take the original descriptions as templates and shorten them to brief descriptions of only one to five words. The selected phrases are still informative enough to distinguish the subtopics of a topic. It is important to note that we were not prejudiced against the cluster labels of the approaches, but solely strive for meaningful and discriminative descriptions for the subtopics. Besides the descriptions, we collect images that represent the subtopics using a web search engine. In case we cannot obtain a suitable image for a subtopic, we

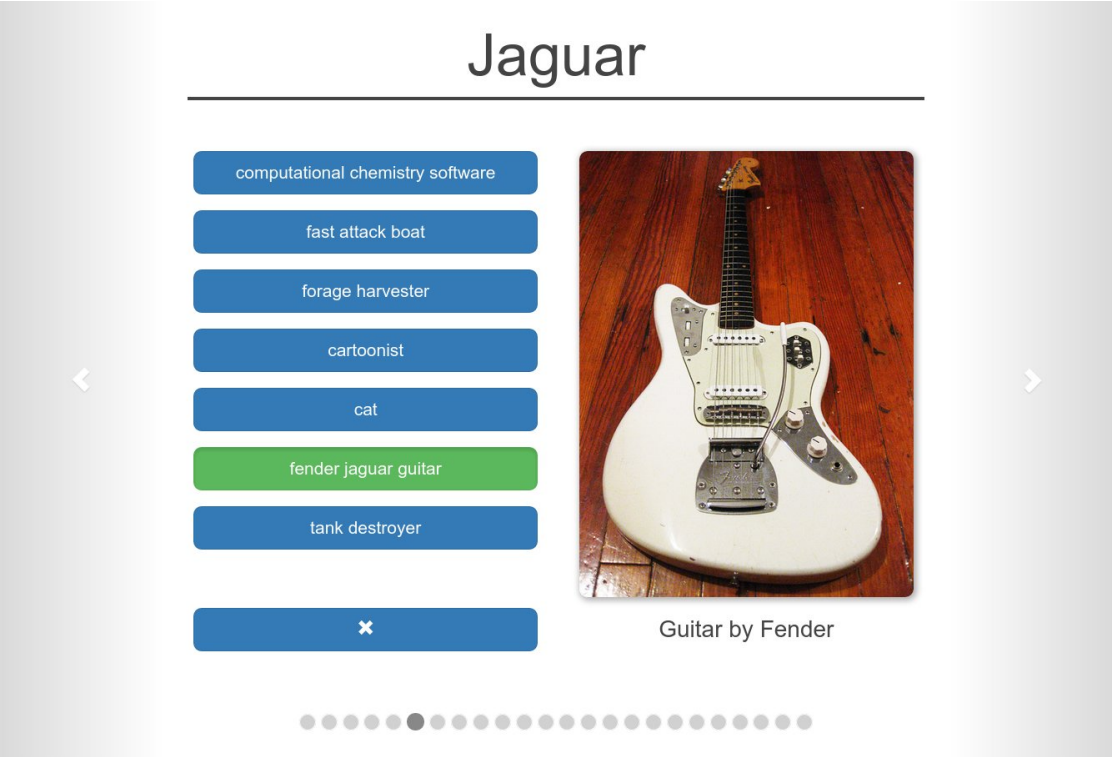

Figure 4.13: Screenshot of the first experiment of the user study, which strives for examining the discriminative power of the cluster labels of our and the two baseline algorithms.

replace this subtopic from the evaluation corpus with the next subtopic that maximizes the average F-measure.

Figure 4.13 shows an example of a topic as it is presented to a participant. The name of the topic, here "Jaguar", is shown at the top, and the cluster labels (i. e., the subtopics unveiled by the approach) are listed at the left-hand side of the screen. When the participant selects or hovers over a label, the label is highlighted with a green color (here, "fender jaguar guitar"). If none of the labels is satisfying, the participant clicks the lowermost button with a cross in it, which is then highlighted with a red color. The given subtopic, for which the best fitting label has to be selected, is presented by its image and short description next to the labels at the right-hand side. The participant can navigate to the previous or next subtopic by clicking the controls at the very left or right, respectively. At the bottom, the participant can see the progress of the experiment and can also navigate to a desired subtopic by clicking one of the indicators.

Table 4.4: Results of the user study on the discriminative power of our approach (leftmost) compared to the baseline algorithms.

Judgment	Constrained k-means	Descriptive k-means	k-means + chi-square
✓	**213**	180	152
×	**15**	25	39
–	**21**	44	58
$\sum$	249	249	249
Accuracy	**0.9825**	0.9670	0.9527
F_1 score	**0.9221**	0.8392	0.7581
p-value	–	0.0049	0.0000

Experimental Process From four different local high schools, 46 German pupils and three teachers (23 female, 26 male, mean age 18.3, $SD = 6.8$) with five or more years of English courses participated in five groups. When a group arrived in our lab, the pupils were randomly assigned to a lab seat. After an introduction to document clustering, we ran a short demo on a projector in order to ensure that the participants are familiar with the GUI and the evaluation process. The 249 judgments for each of the three approaches result in about 15 judgments per participant. We ensure that no participant has to judge the same subtopic twice (neither for different approaches). Moreover, we shuffle the list of labels for each participant. After the participants have judged the last subtopic, another click on the right indicator leads to an intermediate slide that states the end of the first experiment. We did not restrict the time to finish the experiment, and already finished participants could immediately start the second experiment by clicking the right indicator again.

Results and Discussion All participants judged each of their 15 subtopics, so that we collected 249 judgments for each approach. The detailed aggregated numbers on the performance of each approach are listed in Table 4.4. The first row denotes the number of judgments where the selected label constitutes also the label that the approach generated for the given subtopic. In other words, the first row indicates correct assignments and hence true positives. The second row lists the number of judgments where the participant selected a different label than that the approach generated for the given subtopic. Therefore, the selection was wrong and denotes a false positive. The last category is listed in the third row, in which the numbers indicate the judgments where the participant selected neither of the presented labels; this case denotes a false negative. For each category, the number indicating the best performance is listed in

bold. As the individual numbers reveal, our approach performs best in each of the three categories. A common measure to merge the individual numbers is the accuracy, which is the proportion of true results among the total number of cases examined. Note that the accuracy deviation is rather small in consideration of the large deviation of the individual numbers. This is based on the fact that the accuracy not only considers the true positives as a true result but also the true negatives. That is, whenever a participant selected the correct cluster label or none, the remaining (maximum) seven labels denote true negatives. In case that the participant selected a wrong label, the true negatives decrease by one. As a consequence, even the rather moderate performance of k-means plus chi-square results in a high accuracy. A more reliable evaluation is given by the F-measure, which is listed as the F_1 score, since we chose $\beta = 1$ (cf. Section 3.4.3). As the F-measure ignores the true negatives in its calculation, it is a much more reliable measure in our scenario.

To statistically estimate the per-individual effect, we compare the ratio of correct label assignments (true positives) among all assignments given for a subtopic (true positives, false positives, false negatives). Each subtopic is judged by three participants, and the assigned labels split into correct (true positives) and incorrect (false positives and false negatives). In case that all three participants select the correct label, the ratio equals $\frac{3}{3} = 1$. If only one participant decided for the correct label, the ratio is $\frac{1}{3}$. According to the Shapiro-Wilk test [SW65], the individual participants' ratios are not normally distributed for either approach such that we choose a non-parametric significance test [LFH10]. For our within-subjects design with ratio data and three to-be-compared approaches, the Wilcoxon signed rank test [Wil45] is known as a suitable significance test. For the 49 participants' ratios we get a p-value of 0.0049 when comparing the distribution of our approach and Descriptive k-means, and a p-value below 0.0000 compared to k-means plus chi-square. Since both p-values are below 0.05, we can reject the null hypothesis that the ratios' distributions are equal. In other words, our approach significantly increases the discriminative power of the cluster labels when compared to the two baseline algorithms, and we can thus accept our hypothesis, that our algorithm increases the discriminative power of cluster labels.

Experiment 2: Descriptive Power

In the second experiment, we examine the descriptive power of the cluster labels. Our hypothesis is that using our approach increases the descriptive power of the cluster labels when compared to the baseline algorithms.

Experimental Design The experiment is directly attached to the first experiment, so that after the participants have finished the first experiment, they seamlessly continue with the second experiment. Rationale for the experiment is that a participant is confronted with the different cluster labels that are generated by the approaches for

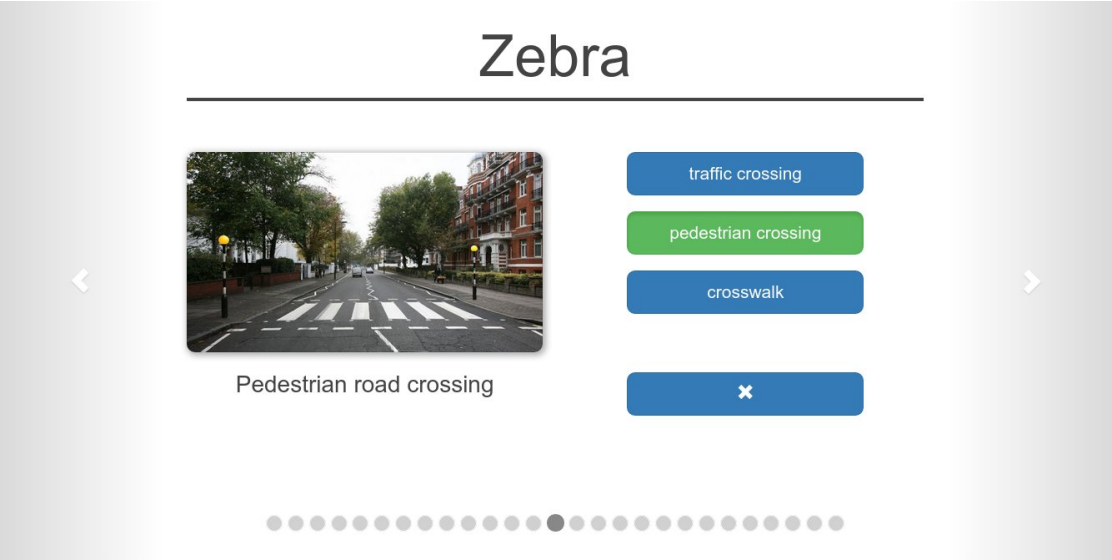

Figure 4.14: Screenshot of the second experiment of the user study, which strives for examining the descriptive power of the cluster labels of our and the two baseline algorithms.

one subtopic, and has to select that label which best describes the given subtopic. We ensure that the clusters of the approaches cover the same subtopic by calculating their F-measures to the subtopic. Only if the cluster of each approach exceeds the threshold of 0.8 with regard to the subtopic documents, we include that subtopic to the data set of this experiment. Thereby, we ensure not only the strong connection of the cluster to the subtopic but also that all three approaches perform equally well for this subtopic. Since a subtopic is represented in the same way as in the first experiment, we further discard those subtopics for which we cannot obtain a suitable image. In summary, we are able to keep 226 of the 468 subtopics, leading to a total of 678 judgments (again, we want each subtopic to be judged by three different participants). In case that two or all approaches have generated the same cluster label, we only present the label once to the participant. Note that the subtopic is only a representative for the cluster documents. Although the descriptive power demands the appropriateness of the cluster label to each of the cluster documents (and not to its abstract topic), we ensured the strong connection between the document and the subtopic when manually selecting the documents for a subtopic in the data set generation in Section 4.1.

Figure 4.14 shows an example of a subtopic as it is presented to a participant. It is very similar to the GUI in the first experiment with only small differences. Again, the name of the topic, here “Zebra”, is shown at the top of the screen. However, this time the relevant subtopic, for which the approaches have generated the cluster label, is shown on the left, which shall visually distinguish the first from the second experiment. The

Table 4.5: Results of the user study on the descriptive power of our approach (leftmost) compared to the baseline algorithms.

Voting	Constrained k-means	Descriptive k-means	k-means + chi-square
✓✓✓	48	45	19
✓✓	52	45	36
✓	51	49	55
–	75	87	116
$\sum_{aggregated}$	**299**	274	184
p-value	–	0.3525	0.0000

different labels are listed next to the subtopic at the right-hand side of the screen. Like in the first experiment, the selected label is highlighted with a green color (here, "pedestrian crossing") whenever a participant selects or hovers over a label. If none of the labels is satisfying, the participant clicks the lowermost button with a cross in it, which is then highlighted with a red color. The navigation controls are still on the very left or right, respectively. The indicators at the bottom denote the overall progress of the experiment, so they include also the slides from the first experiment.

Experimental Process The participants are still the same pupils and teachers as in the first experiment, since they simply continue with the second experiment after finishing the first one. The 678 judgments are distributed to the 49 participants, resulting in about 14 judgments per participant. In doing so, we ensure that no participant is confronted with the same subtopic twice. Moreover, we shuffle the list of labels for each participant, so that there is no approach that is treated preferentially. As in the first experiment, we do not restrict the time to finish the experiment.

Result and Discussion All participants judged each of their 14 subtopics, so that we could collect all of the 678 judgments for the subtopics. The detailed aggregated numbers on the performance of each approach is listed in Table 4.5. The first four rows denote the number of judgments where either all three, two, one or none participant(s) voted for the corresponding approach. For all three approaches, the numbers accumulate to the 226 judged subtopics. As the numbers reveal, the participants' overall agreements in judging the cluster labels of our approach and Descriptive k-means are very similar. Nevertheless, the vote distribution is not as distinct as expected. While all three participants voted for our approach in 48 subtopics, in almost the same number of subtopics only one participant voted for our approach and two participants voted for one of the baseline

algorithms (or for none at all). In contrast, the votes for k-means plus chi-squared are not as equally distributed as the other two approaches. In the majority of subtopics, the participants agreed in their votes, i. e., the other approaches generated more suitable labels. In order to obtain all the votes given for each approach, we multiply each row with the corresponding number of votes, and accumulate these products. For example, the sum for our approach is calculated by $48 \cdot 3 + 52 \cdot 2 + 51 \cdot 1 + 75 \cdot 0 = 299$. Note that aggregating the sum of each approach does not lead to the 678 given judgments, since two or all approaches obviously generated the same cluster label for certain subtopics. In case that the participant selected that common label, each of the approaches gets a vote. The cluster labels of our approach are selected in most of the subtopics, which means that our approach again performs better than the baseline algorithms.

Similar to the first experiment, we statistically estimate the per-individual effect. Therefore, we compare the vote distribution for each subtopic of our approach with the vote distribution of the baseline algorithms. More precisely, if two participants voted for the label of our approach and one participant considered the label of Descriptive k-means as most suitable, the vote distribution for that subtopic is two votes for our approach, one for Descriptive k-means, and zero for k-means plus chi-square. Again, we apply the Shapiro-Wilk test [SW65] to examine whether the individual vote distributions for either subtopic are normally distributed or not. Since the test shows that they are not normally distributed, we again choose the Wilcoxon signed rank test as an non-parametric significance test [Wil45]. The p-value of our approach compared to Descriptive k-means is higher than 0.05, and we can thus not reject the null hypothesis that the vote distributions are equal. In other words, our approach does not significantly increase the descriptive power of the cluster labels. However, when comparing the vote distributions of our approach and k-means plus chi-square, the resulting p-value is far below 0.05, so that our approach significantly increases the descriptive power of the cluster labels. Therefore, we can only partly accept our hypothesis.

In conclusion, the comparison of the three approaches reveals that we introduced a competitive approach that partly outperforms the baseline algorithms. While the found clusters and their documents are of comparable high quality for each of the algorithms, the evaluation of the corresponding cluster labels shows that there is a great diversity in their explanatory power. The labels generated by our approach are of significantly higher discriminative power, leading to an increased separability of the found clusters in comparison to the clusters of the baseline algorithms. With regard to the descriptive power of the cluster labels, our approach significantly outperforms one of the baseline algorithms and has a comparable high descriptive power compared to the second baseline algorithm.

5 Conclusion

In this thesis, we revisited the document clustering problem from an information retrieval perspective that explicitly addresses the need for appropriate cluster labels. In particular, users engaged in exploratory search tasks, where the objective is to learn about an unfamiliar topic and to discover new information, can benefit from a document clustering in which they can anticipate the content of a cluster from its label.

The task of grouping documents with high similarities in the same cluster has been a target of research for a considerable long time and many studies confirm the clustering algorithms' effectiveness for that task. However, the quality of the cluster labels was mostly neglected. The situation changed when people realized that clusters could be used as a browsing interface in order to explore and organize a set of unstructured documents, for example websites of the World Wide Web. Chapter 2 in this thesis examined various approaches of the broader spectrum of labeled document clustering and pointed out the major differences to our approach.

In our opinion, the need for a meaningful cluster label goes hand in hand with the grouping of semantically similar documents. With the absence of a descriptive label, users are forced to investigate each cluster for its covering topic and to sift through the cluster documents in time-consuming efforts. An improvement in the document representation is not accomplished, irrespective of the clustering's quality. Chapter 3 introduced our approach towards a labeled document clustering. First, we revealed the strong connection between cluster labels and search queries and could therefore propose the use of a search engine to model the user in the decision of whether a document is relevant for a given cluster label or not. We further suggested to think of cluster labels as search queries that have to retrieve the documents of the associated cluster. The generation of suitable queries was guided by the three main characteristics of meaningful cluster labels depicted by Stein and Meyer zu Eißen [SM04]: (1) the comprehensive power, (2) the descriptive power and (3) the discriminative power. In order to adopt these characteristics in our approach, we introduced several constraints that were integrated in certain steps of our processing pipeline. Each step was evaluated using an extended version of the Ambient dataset—the Ambient++ data set, which includes 4,680 manually annotated documents, and which will be publicly available.

As a first step, we generated an appropriate vocabulary that formed the basis from which the search queries were built. All methods satisfied the query-syntax constraint, which ensured the comprehensive power of the queries by only allowing the extraction of noun phrases. According to several evaluations in Section 4.2, including the introduction

of a novel evaluation measure, which we called Soft F-measure, we decided to extract six noun phrases of each document with the help of natural language processing. These noun phrases further had to satisfy the query-length constraint, which limited the phrases to a maximum length of four terms.

In the second step, we indexed the documents by means of the generated vocabulary using a search engine. This way, we obtained the relevant documents for each query, which later served as labeled cluster candidates. We investigated different retrieval models, and with regard to the evaluation in Section 4.3, we decided to incorporate a topic model, namely Explicit Semantic Analysis, in the retrieval process of the search engine. With the implementation of one of the relevance constraints, we were able to restrict the result list to the most relevant documents. The evaluation showed that the best results for our scenario were achieved using the top-10 constraint, which cuts the result list at rank 10. Using an appropriate retrieval model along with a restricting relevance constraint ensured the descriptive power of the queries (and in turn, the cluster label candidates), whereby the labels should speak for each document in a cluster.

The retrieving queries for each document (called its keyqueries) then served as features for a cluster analysis. We developed three clustering algorithms that all satisfied the common-query constraint, which required the documents of a cluster to share at least one common keyquery. The best performing algorithm was Constrained k-means, which was modeled on the popular k-means algorithm. The found queries for the clusters served as cluster labels, while their corresponding result lists served as cluster documents. By introducing the common-query constraint, the discriminative power of cluster labels was guaranteed. Each label has to discriminate its cluster documents from documents in other clusters, since similar documents would have been retrieved for that labeling query as well and therefore grouped in the same cluster.

Once the best processing pipeline of our approach was determined, we measured its effectiveness in comparison to two state-of-the-art algorithms: Descriptive k-means and k-means plus chi-square. The evaluation process in Section 4.5 was twofold: (1) We first determined to what extent the algorithms were able to group similar documents in the same cluster. As the evaluation revealed, the algorithms achieved almost comparably high performances. Since the grouping of similar documents is only one criterion towards a good document clustering, we (2) also evaluated the quality of the corresponding cluster labels and their explanatory power. We therefore conducted a user study involving 49 participants. The first part of the experiment was aimed at accessing the discriminative power of the cluster labels. We therefore presented the participants the generated labels for each subtopic of 22 topics from the Ambient++ data set and asked them to select the most suitable label for a given subtopic. The user study showed that our Constrained k-means algorithm could significantly increase the discriminative power of cluster labels. The second part of the experiment investigated the descriptive power of cluster labels. After presenting the participants the labels of each algorithm, we asked them to select that label which best described a given subtopic. While our

algorithm and Descriptive k-means produced labels of a similar descriptive power, we could significantly outperform the second baseline algorithm, k-means plus chi-square.

In conclusion, we presented a novel approach towards labeled document clustering that achieved a comparably high performance with regard to the cluster documents and a performance increase concerning the exploratory power of the cluster labels in comparison to the baseline algorithms. However, for our novel view on document clustering from an information retrieval perspective, the thesis at hand only laid the ground for further valuable research.

With respect to our processing pipeline, we consider it worthwhile to include a predefined taxonomy in the vocabulary generation step. Although we are confident that meaningful cluster labels are also contained in the cluster documents, an independent taxonomy could ensure appropriate vocabulary terms and reduce the size of the vocabulary. As a starting point, we could consider our manual subtopic descriptions as a ground truth taxonomy and investigate to what extent the Best Clustering covers the corresponding subtopic documents. If it was possible to achieve this task with a high accuracy, taxonomies from classification systems like the Open Directory Project[1] or from Wikipedia could be used as the vocabulary. In this regard, it is advisable to evaluate the different vocabulary generation methods with the help of a user study for a more reliable interpretation than it is possible for the Best Clustering. Further effort should also be put into the implementation of a more suitable relevance constraint. In case of a known number of desired cluster documents, the top-k constraint achieved good results. However, the score constraint as a first approach towards a variable cut-off rank might still leave room for improvement; therefore, it requires a detailed investigation on the interplay between the retrieval scores of the search and the cut-off rank of the score constraint. Moreover, the evaluations in this thesis were exclusively based on the Ambient++ data set. Although this data set is an extension of the popular Ambient data set, the findings should be confirmed using an alternative data set. Therefore, we again suggest the incorporation of the Open Directory Project.[1]

There are numerous application areas for labeled document clustering. With our processing pipeline consisting of exchangeable approaches for each step, we strive for a seamless integration into search interfaces, which simultaneously provides a cost-efficient maintenance. In this regard, we consider it worthwhile to investigate the capability for a hierarchical clustering of documents. As a starting point, we could consider all documents of the 44 topics of the Ambient++ data set as the first document collection to be clustered. After a user selected one of the discovered topics, the system then goes one step down the hierarchy and unveils the corresponding subtopics of the selected topic by considering its documents as the next document collection to be clustered.

[1] `http://www.dmoz.org`, Last accessed: February 13th, 2015

List of Figures

List of Tables

Bibliography

[ABDR06] Eugene Agichtein, Eric Brill, Susan Dumais, and Robert Ragno. Learning user interaction models for predicting web search result preferences. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 3–10, New York, NY, USA, 2006. ACM.

[AS09] Maik Anderka and Benno Stein. The ESA retrieval model revisited. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 670–671, New York, NY, USA, 2009. ACM.

[BC00] Ken Barker and Nadia Cornacchia. Using noun phrase heads to extract document keyphrases. In *Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence*, AI '00, pages 40–52, London, UK, 2000. Springer.

[BCDG08] Francesco Bonchi, Carlos Castillo, Debora Donato, and Aristides Gionis. Topical query decomposition. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 52–60, New York, NY, USA, 2008. ACM.

[BDW08] Sugato Basu, Ian Davidson, and Kiri Wagstaff. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman & Hall/CRC, London, UK, 2008.

[BJR08] Cory Barr, Rosie Jones, and Moira Regelson. The linguistic structure of English web-search queries. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 1021–1030, Stroudsburg, PA, USA, 2008. ACL.

[CKPT92] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '92, pages 318–329, New York, NY, USA, 1992. ACM.

[CORW09] Claudio Carpineto, Stanislaw Osiński, Giovanni Romano, and Dawid Weiss. A survey of web clustering engines. *ACM Computing Surveys*, 41(3):17:1–17:38, 2009.

[CRZ09] David Carmel, Haggai Roitman, and Naama Zwerdling. Enhancing cluster labeling using Wikipedia. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 139–146, New York, NY, USA, 2009. ACM.

[DDF+90] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the Association for Information Science*, 41(6):391–407, 1990.

[FG04] Paolo Ferragina and Antonio Gullì. The anatomy of SnakeT: A hierarchical clustering engine for web-page snippets. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Knowledge Discovery in Databases: PKDD '04*, volume 3202 of *Lecture Notes in Computer Science*, pages 506–508. Springer, 2004.

[FLGD87] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971, 1987.

[FLSG12] Norbert Fuhr, Marc Lechtenfeld, Benno Stein, and Tim Gollub. The optimum clustering framework: Implementing the cluster hypothesis. 15(2):93–115, 2012.

[GHMS13] Tim Gollub, Matthias Hagen, Maximilian Michel, and Benno Stein. From keywords to keyqueries: Content descriptors for the web. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 981–984, New York, NY, USA, 2013. ACM.

[GM07] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1606–1611, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[GVHS14] Tim Gollub, Michael Völske, Matthias Hagen, and Benno Stein. Dynamic taxonomy composition via keyqueries. In *Digital Libraries 2014: 14th ACM/IEEE Joint Conference on Digital Libraries (JCDL '14), 18th International Conference on Theory and Practice of Digital Libraries (TPDL '14)*, London, UK, 2014. ACM/IEEE.

[JD88] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[JGP+05] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 154–161, New York, NY, USA, 2005. ACM.

[KFN10] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. Boilerplate detection using shallow text features. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 441–450, New York, NY, USA, 2010. ACM.

[LA99] Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 16–22, New York, NY, USA, 1999. ACM.

[LFH10] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. *Research Methods in Human-Computer Interaction*. Wiley Publishing, Indianapolis, IN, USA, 2010.

[MC07] Rada Mihalcea and Andras Csomai. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 233–242, New York, NY, USA, 2007. ACM.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[MS99] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999.

[OSW04] Stanisław Osiński, Jerzy Stefanowski, and Dawid Weiss. Lingo: Search results clustering algorithm based on singular value decomposition. In *Intelligent Information Processing and Web Mining*, IIPWM '04, pages 359–368. Springer, 2004.

[PBR07] David Pinto, José-Miguel Benedí, and Paolo Rosso. Clustering narrow-domain short texts by using the Kullback-Leibler distance. In *Proceedings of the 8th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing '07, pages 611–622, Berlin, Germany, 2007. Springer.

[PCG10] Jeremy Pickens, Matthew Cooper, and Gene Golovchinsky. Reverted indexing for feedback and expansion. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1049–1058, New York, NY, USA, 2010. ACM.

[Pea00] Karl Pearson. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175, 1900.

[Rij79] Cornelis Joost van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 1979.

[RZ09] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

[SFMC12] Ugo Scaiella, Paolo Ferragina, Andrea Marino, and Massimiliano Ciaramita. Topical clustering of search results. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 223–232, New York, NY, USA, 2012. ACM.

[SGH11] Benno Stein, Tim Gollub, and Dennis Hoppe. Beyond precision@10: Clustering the long tail of web search results. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 2141–2144, New York, NY, USA, 2011. ACM.

[SM04] Benno Stein and Sven Meyer zu Eißen. Topic identification: Framework and application. In *Proceedings of the International Conference on Knowledge Management*, volume 400 of *I-KNOW '14*, pages 522–531, 2004.

[ST09] Giovanni Maria Sacco and Yannis Tzitzikas. *Dynamic taxonomies and faceted search: theory, practice, and experience*. Springer, New York, NY, USA, 2009.

[SW65] Samual S. Shapiro and Martin. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52:591–611, 1965.

[SWY75] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[TC06] Pucktada Treeratpituk and Jamie Callan. An experimental study on automatically labeling hierarchical clusters using statistical features. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 707–708, New York, NY, USA, 2006. ACM.

[Vaz01] Vijay V. Vazirani. *Approximation Algorithms*. Springer, New York, NY, USA, 2001.

[Wei06] Dawid Weiss. *Descriptive Clustering as a Method for Exploring Text Collections*. PhD thesis, University of Technology, Poznan, Poland, 2006.

[Wil45] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83, 1945.

[WZ07] Xuanhui Wang and ChengXiang Zhai. Learn from web search logs to organize search results. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 87–94, New York, NY, USA, 2007. ACM.

[ZE98] Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 46–54, New York, NY, USA, 1998. ACM.

[ZHC+04] Hua-Jun Zeng, Qi-Cai He, Zheng Chen, Wei-Ying Ma, and Jinwen Ma. Learning to cluster web search results. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 210–217, New York, NY, USA, 2004. ACM.