Dynamic Taxonomy Composition via Keyqueries

Tim Gollub Michael Völske Matthias Hagen Benno Stein
Bauhaus-Universität Weimar
99421 Weimar, Germany
<first name>.<last name>@uni-weimar.de

ABSTRACT

This paper presents an unsupervised framework for dynamic, subject-oriented taxonomy composition in digital libraries, which can naturally integrate existing library classification systems. The taxonomy classes in our approach correspond to so-called keyqueries that are run against the digital library's full-text retrieval system. Given a document, a keyquery is a set of few keywords for which the document achieves a high relevance score. Keyqueries can hence be viewed as a general and concise description of the returned retrieval results. The keyquery framework addresses important problems of static classification systems: overlarge classes and overly complex taxonomy structures. If, for instance, a leaf class grows to an indigestible size, keyqueries for the contained documents provide a suitable split mechanism. Since queries are well-known to library users from their daily web search experience, they increase the structural complexity in a transparent way.

The paper presents also a strategy for taxonomy-based library exploration. Given a user's information need in the form of library documents, we synthesize a hierarchy of keyqueries that covers this library subset. We manage to solve this difficult set covering problem on-the-fly by combining inverted and reverted indexes along with heuristic search space pruning within a map-reduce application. An empirical evaluation with an ACM collection of scientific papers demonstrates the efficiency and effectiveness of our taxonomy composition framework.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]: Retrieval Models, Query Formulation

General Terms: Algorithms, Experimentation, Performance

Keywords: dynamic taxonomy composition, keyquery, classification systems, reverted index, big data problem

1. INTRODUCTION

A classification system or taxonomy consists of a set of classes and a set of instances, where each instance is assigned to or tagged with some of the classes. In a hierarchical classification system, classes are usually arranged to as a class tree, where the root is divided into subclasses until leaf classes are reached.[1] In this paper we use the term "taxonomy" to refer to all kinds of hierarchical classification systems that employ such a class tree. We present a framework for the unsupervised composition and maintenance of subject-oriented taxonomies for digital libraries.

[1]The Colon Classification is a noteworthy exception.

Subject-oriented classification systems have a long tradition in libraries as a tool to find and locate relevant documents. The today's most widely used systems, the Dewey Decimal Classification and the Library of Congress Classification, look back on a history of successful application spanning more than a century [25]. However, since the digital age hit the library world in the late 20th century, the usefulness of maintaining a subject-oriented classification system for the users is not that obvious anymore. Digital content renders shelves obsolete, and, with them, the indispensable need for a location encoding based on subjects. Furthermore, querying a search engine that indexes the documents' meta-data and full-texts turns out to be more effective for identifying an initial set of relevant library documents than browsing a classification system [24]. Given this situation, we ask if and in which form a classification system could be beneficially applied in a digital library.

1.1 Use Case Study

The following paragraphs present four use cases that go beyond the identification of relevant documents; they discuss how a taxonomy or classification system can complement the capabilities of a query-based search engine. The four use cases are (a) the recommendation of documents on the basis of given ones, (b) the appreciation of serendipities, (c) the exploration of large information spaces, and (d) the profiling of entities.

(a) Document Recommendation. In a document recommendation setting, users have an incomplete set of relevant documents at hand (e.g., references in a paper or search results for a specific subject), and want to complement this set with other relevant documents from the library. A conventional query-based search engine is not really capable to support this user need, since its inputs are keywords and not document sets. Most existing document recommender systems are based on citation information for the documents, from which they infer semantic relations. In addition, and in the general case where no citation graph is given, some measure of topical similarity is used to identify documents which address a common subject [33]. To this end, a reasonable approach for document recommendation is to identify the most specific set of classes which cover the given documents. The recommendation can then be comprised of the other documents from these classes. We refer to the most specific classes covering a set of documents as *complementation classes*.

(b) Serendipity Search. In a serendipity search setting, users already know the documents they want to retrieve, but appreciate the discovery of other interesting documents. Serendipities are a well-known phenomenon in physical libraries that employ a subject-

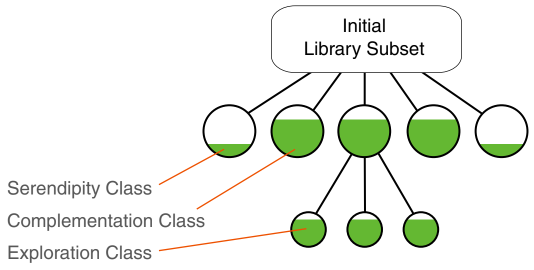

Figure 1: Conceptual illustration of our dynamic taxonomy composition approach. An initial library subset is subdivided by the means of complementation, exploration, and serendipity classes. The filling level of the classes denotes the precision of a class with respect to the initial library subset.

oriented location encoding. On their way to the books of interest, users pass book shelves on related subjects, allowing some of the books to catch their attention. In a digital environment, designing for serendipity is centered around the presentation of visual clues that trigger the formation of a new information need in the user's mind [27]. If we assume a user who submits a query to a search engine, a possibility to foster serendipities would be the presentation of search results which are related to the original information need. However, the top entries of a search result list are very precious, and devoting them to serendipity candidates seems not a good idea. On the other hand, serendipity candidates put somewhere in the result list's tail would just get lost in the shuffle. A more reasonable approach is to find classes which are semantically close to the subjects of the search results, and to present those classes next to the results. We refer to such semantically close classes as *serendipity classes*.

(c) Exploratory Search. In an exploratory search setting, users are faced with a large set of documents and want to learn about their contents [32]. The given document set may be the result set of a broad query, the combined results of a longer search session, or be provided by a third party (e.g., in an e-discovery setting). If the documents are presented as a long list, it is hard for users to get an overview of the collection. More likely, they just lose track. In this respect, studies on search behavior reveal that result lists are examined only rarely beyond the top results, and that only 8% of users ever click on results beyond rank 30 [24]. To provide a structured access to the documents, a taxonomy can be composed that iteratively subdivides the given documents into more specific subtopics, until subclasses of manageable sizes are reached(typically, < 30 documents).[2] The subclasses then serve as reference points users could use to infer higher level concepts and to work out an exploration strategy. We refer to subclasses which subdivide the complementation and serendipity classes as *exploration classes*.

(d) Profiling. In a profiling setting, users have a document set which refers to a specific entity, and are interested in obtaining a conceptual map of the entity's contributions. In the context of digital libraries, users may be interested in profiling authors, journals, or the library as a whole. A popular visualization of profiles are word clouds, where the relevance of a subject is shown in terms of font size or color. As the *profile classes* of an entity from the library, the complementation classes and the exploration classes can be used; rendering profiling as special case of an exploratory search setting.

A common theme of the above described four use cases is that users are equipped with an initial set of documents (e.g., by querying the library's full-text search engine), and benefit from classes which are tailored to this initial set. For taxonomies or classification systems in digital libraries, this indicates that the user's need for a single one-fits-all classification system, which lets her browse for relevant documents, is not the dominant task anymore. This task, once driving library classification research, is almost completely subsumed by the provision of decent query-based search engines. Instead, the task important for users is to provide a taxonomy consisting of complementation, serendipity, and exploration classes tailored to a given but arbitrary library subset—the user's information need. We illustrate this concept in Figure 1. Stated more formally, the Aristotelean classification philosophy that distinguishes essential from accidental document properties, and which strives for one classification of literature along its natural joints, should be complemented by a more holistic philosophy in the vein of the Web 2.0 movement—to include and postpone [31]. Including every class that is potentially useful to describe a library subset, and postponing the instantiation of the classification system to the moment where the information need of a user is known.

1.2 Keyqueries as Taxonomy Classes

There are two principal approaches to compose a taxonomy for a given set of documents: (1) applying a hierarchical clustering algorithm to the documents and labeling the resulting clusters, and (2) compiling a set of candidate classes with labels and document assignments, and composing a taxonomy for the given documents on their basis. With respect to the first option, the unsolved cluster labeling problem prevents an effective application to our use cases (cf. Section 2 for a more detailed discussion on the issues of document clustering). We thus turn to the second option—dynamic taxonomy composition with a given set of classes. To obtain a set of classes with labels and document assignments, we identify the library's query-based search engine as a powerful and flexible choice. We exploit the "wisdom" of the search engine by taking as classes the set of queries that can be formulated on the basis of the search engine's vocabulary, and as class members their respective search results. In other words, we apply the keyquery concept to dynamic taxonomy composition [11].

The use of the library's search engine as the source for classes has two big advantages compared to the use of other knowledge bases, but also one caveat that has to be handled in a reasonable way.

The first advantage is that both systems, the search engine and the taxonomy composition framework, are always in sync. Whenever the search engine indexes a new document, the taxonomy composition will instantly feature the new document without the need for an additional integration process. If a document introduces a new subject, the set of classes is dynamically extended with the new subject. In this respect, we can think of the search engine as an implicit tagging mechanism which tags each document with its keyqueries. This automated tagging mechanism introduces a great deal of flexibility to the library management. Since the documents are automatically integrated and annotated with classes, human intervention can concentrate on documents for which no reasonable taxonomy can be composed. In those cases, library experts can formulate appropriate classes, assign them to the documents, and force a re-indexing to commit the changes to the search engine (and hence to the taxonomy composition framework).

The second advantage of our approach is that search engine theory provides a well-defined concept for the integration of multiple knowledge resources into a common representation, as well as for the combination of queries to a joint class: retrieval models. A retrieval model is the formalization of a linguistic theory that describes how to quantify the relevance of a document for a query, or, in our

[2]The Open Directory Project guidelines suggest to already subdivide a class if it exceeds 20 documents.

view, for a class. In sophisticated search engines, hundreds of features influence the relevance computation, and machine learning is used to find the optimum feature weights [3]. There is an enormous amount of research that has been dedicated to the development of retrieval models for search engines, and this knowledge is now at our disposal. For instance, a retrieval model could be employed which indexes the existing meta-data about the documents with learned feature weights [22]. The retrieval model could itself classify the documents into a hierarchical classification system [14], or tag the documents according to a vocabulary of subject headings [29]. Even further, sophisticated retrieval models employ linguistic resources and thesauri like WordNet[3] or lexical taxonomies like Probase[4] to infer further classes through synonym and hypernym relationships [16].

Though the presented advantages are appealing, there is one issue with the use of search engines in taxonomy generation. For a reasonable use of queries and their search results as classes, inconsistencies with the common perception of classes in a library classification system have to be fixed. To this end, we introduce five constraints for the dynamic composition of taxonomies on the basis of queries (cf. Section 3 for more details). Two class label constraints ensure that the queries used have the look and feel of conventional class labels, one class assignment constraint ensures that only those documents are assigned to a class for which relevance is substantial, and two class composition constraints control the taxonomy's structure and introduce the notion of class generality.

1.3 Taxonomy Composition

Let D_u denote a set of documents and $\mathcal{Q}$ denote a set of candidate queries that adhere to the five constraints. The problem of taxonomy composition for D_u using classes from $\mathcal{Q}$ can then be stated as an optimization problem that has to be solved iteratively until leaf-classes are reached:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{|\mathcal{Q}|} x_i \, |D_i \cap D_u| \\ \text{subject to} \quad & \sum_{i=1}^{|\mathcal{Q}|} x_i \leq k, \\ \text{where} \quad & x_i \in \{0,1\}. \end{aligned}$$

The objective is to find in each iteration the k-subset of $\mathcal{Q}$ that maximizes the recall with respect to D_u, subject to the constraint that the maximum fan-out k is never exceeded. The sets D_i in the formula refer to the documents which are assigned to the class $\mathcal{Q}_i$, and the x_i are indicator variables where a value of one denotes that class $\mathcal{Q}_i$ is included in a particular k-subset. In Section 3.2, we present a time efficient greedy set-cover algorithm for this optimization problem.

In Section 4, we report on a series of experiments that demonstrate the feasibility of our approach for a digital library of 30,000 scientific papers. In the first experiment, we study the runtime characteristics of our approach by simulating a library which grows over the years. Our results show, that our approach can serve as an online library tool that dynamically composes a taxonomy for initial library subsets up to sizes of several thousand documents. In the second experiment, a taxonomy is computed for the whole library, and its profile is compared to a hand-crafted reference taxonomy for the library. In a third experiment, the effectiveness of dynamic taxonomy composition for document recommendation is studied. Compared to the reference taxonomy, our approach yields much higher recommendation precision while keeping recall at an acceptable rate.

Our scientific contributions are fourfold. (1) We suggest dynamic taxonomy composition for various real-life use cases of classification systems in digital libraries (cf. the introductory discussion). (2) We suggest the retrieval model of query-based search engines as a powerful and flexible implicit tagging mechanism that can be used to infer a rich set of diverse classes for the library documents (cf. Section 3). (3) We present a greedy set-cover algorithm for the iterative composition of query-based taxonomies for arbitrary library subsets (cf. Section 3.2). (4) We demonstrate the efficiency and effectiveness of our approach in a case study with 30,000 scientific papers (cf. Section 4).

Related scientific literature is discussed in the following section.

[3]`http://wordnet.princeton.edu/`

[4]`http://research.microsoft.com/en-us/projects/probase/`

2. RELATED WORK

In this section, we review research related to our dynamic taxonomy composition framework from the broader spectrum of information systems that provide a combination of querying and browsing facilities to their users. We describe the various approaches, point out the major differences to our approach, and give references to the state of the art. For a general introduction to classification theory and historical background information, we refer the interested reader to Arlene Taylor's "The Organization of Information" [25]. An excellent discourse about the Web 2.0 and its implications for digital libraries is given by David Weinberger in his book "Everything is Miscellaneous" [31].

The spectrum of information systems we consider is spanned by search engines on the one end, and hierarchical classification systems on the other. Between these boundaries, there are extensions of the systems at either extreme that add browsing or querying facilities to their basis. In the following paragraphs, we traverse the spectrum of information systems from systems conceptually closer to query-based search engines to those that are closer to classification systems.

Diversified Query Suggestions. In most modern search engines for the Web, the user is provided a list of up to ten query suggestions (or auto-completions) while typing a query into the search box. The query suggestions are usually mined from the search engine's query log (Wikipedia topics constitute an alternative source [13]), and contain the most popular queries that are syntactically similar to the entered query artifact. To increase the probability of a relevant suggestion, diversification of query suggestions has been proposed recently [13, 17, 23]. The aim of diversification is to compose a list of syntactically similar queries that are semantically diverse. Diversified query suggestions can hence be interpreted as a dynamically composed, flat classification system for the given query artifact. In contrast to our approach, candidate classes are not derived from the query artifact's search results, but rather from its character sequence. Through this difference, the two systems support different use cases and complement each other: Query suggestions help users find an initial query, while our approach provides a hierarchical structuring of the results once retrieved.

Diversified Query Recommendation. Query recommendation refers to the task of providing, next to the search results, a list of queries that are semantically equivalent to the user's submitted query. Semantic equivalence is typically measured by click through logs. From the set of candidate recommendations, the most popular (= most frequent in the query log) are chosen for presentation [2]. The rationale behind query recommendation is to educate the user about the "better" or more common ways of expressing her information need. As for query suggestions, diversification has been proposed also for query recommendations. Li et al. cluster queries into so-called query concepts (i.e., query groups of high semantic equivalence), and draw a single query from each concept as candidate recommendation to compile a diversified, non-redundant query recommendation list [15]. This kind of diversification is most useful for ambiguous queries, where diverse recommendations allow users to pick a query that specifies their information need more precisely. The problem of dynamic taxonomy composition is related, in that it can also be used to add a list of queries next to the search results. However, dynamic taxonomy composition strives for queries dividing the search results into more specific subclasses, and hence is best applied to non-ambiguous queries (for exploration, recommendation, and serendipities).

Query-Based Taxonomy Composition. The task of composing a taxonomy for a given set of documents falls into two principal steps: the acquisition of classes and their arrangement to a taxonomy. For the first step, the majority of existing taxonomy composition approaches employ keyphrases extracted from the documents as the class set [16, 19, 21]. Liu et al. submit individual keyphrases as queries to a search engine, but solely for the purpose of building a bag-of-words representation of the keyphrase classes [16]. The use of queries as classes—as we propose it—can be regarded as an extension to the keyphrase strategy. Our framework comes with two advantages: First, the combination of keyphrases to joint classes has a well-defined interpretation: the underlying conjunctive query. Second, employing a topic model like ESA [10], it is possible to find keyphrases that do not explicitly appear in a document's text.

Queries have been considered as classes by Chuang et al. [7] and Bonchi et al. [5]. In their work on query clustering, Chuang et al. use the query log of a search engine for class acquisition, and apply a multi-branch hierarchical clustering algorithm to compose a taxonomy for the queries. We see the clustering approach as a bottom-up alternative for the greedy set-cover algorithm we employ for taxonomy composition. A bottom-up strategy is efficient, when the task is to include the whole class set into the taxonomy. However, in our scenario, where only a subset of the classes is chosen for the taxonomy, a top-down approach like ours is much more efficient. A greedy set-cover algorithm similar to ours is employed by Bonchi et al. in their work on query decomposition, but with different constraints. While we strive for an iterative decomposition into gradually more specific queries on every level, the goal of Bonchi et al.'s query decomposition is a flat classification system of queries that have small overlap and maximum precision, irrespective of their generality. A further difference is that topical query decomposition starts from an initial query, while our approach starts from a given library subset for which a single query might not exist.

As an alternative for the second principal step—taxonomy composition—Navigli et al. [19] mine hypernym relationships from the documents' texts to build up a graph of class relationships, and prune the graph to a tree in a post-processing step. Thanks to the flexibility of the retrieval models behind query-based search engines, our framework can integrate an approach like that: the retrieval model can simply assign to the library documents all known hypernym relationships. A query with a hypernym then retrieves the appropriate documents and can also be used by our greedy set-cover algorithm.

Document Clustering. Document clustering is the unsupervised complement to classification. It aims at grouping similar documents into one cluster. The objective of document clustering hence aligns well with the objective of dynamic taxonomy composition. The best known approach to document clustering is Scatter/Gather [8], which, like our approach, creates a cluster (class) hierarchy in an iterative process. In the Scatter phase of the two-step approach, documents are clustered by some cluster algorithm, and the resulting clustering is presented to the user. The user then selects the clusters of interest in the Gather phase, and the documents that belong to the selected clusters are used as input for the next iteration. This unsupervised discovery of classes has the potential to reveal semantic concepts that have no representation in the query space. But there are a couple of drawbacks often preventing document clustering from being effective. The most severe issue of clustering is that the problem of finding a meaningful label, from which the major cluster characteristics can be inferred, is not sufficiently solved [24].

Recent document clustering approaches (cf. [6] for an overview) thus turn to monothetic clustering, producing cluster labels from the terms that appear in each document of a cluster. However, we argue that this approach trades the discovery of sophisticated concepts against an ineffectual use of queries as labels. Furthermore, clustering acts only locally on the given document set and is not directly connected to the underlying search engine. Hence, no additional library documents can be efficiently retrieved for a cluster, preventing complementation and serendipities. Finally, clustering constitutes a shift in the interaction paradigm that is hard to communicate to the user: Clicking on a cluster label reveals documents different from a search for the cluster label. Our proposed dynamic taxonomy composition sidesteps these issues by considering only classes with a representation in the query space. We also suggest the implementation of a feedback loop into our approach that drives the (manual) annotation of documents for which no proper taxonomy can be found.

3. DYNAMIC TAXONOMIES

In this section, we formalize our approach and explain how to compute a keyquery-based taxonomy for a set of library documents. Our proposed strategy is outlined in Figure 2. The upper part shows the class acquisition process: tagging the library documents with their keyqueries. This process runs offline and generates the queries used as candidate classes in taxonomy composition. The lower part of the figure illustrates the online taxonomy composition process. Each level of the taxonomy is computed by solving the optimization problem introduced in Section 1.3. The computation of subsequent taxonomy levels is triggered by the user, who selects a query from the current taxonomy level. We present the two parallel processes in detail in the following, including five constraints that control the various subroutines.

3.1 Class Acquisition

To compose a taxonomy for a set of library documents, a set of classes to draw from has to be acquired. Our approach derives this class set from the query-based search engine, a component commonly available in modern digital libraries [12]. To facilitate the retrieval of documents through queries, the search engine extracts the vocabulary V from the library documents' meta-data and full text, and then indexes the library documents D with respect to V (left column in Figure 2). In the resulting data structure, the inverted index μ, each phrase in V constitutes a key pointing to a postlist that contains all documents with a non-zero retrieval score for that phrase. If a query $Q \subset V$ is submitted to the search engine, the search engine takes each term from Q, collects the documents in the respective postlists, and ranks them according to their aggregated retrieval scores (both sum and product are common for the aggregation).

For the computation of retrieval scores, a large variety of retrieval models are available [18]. These range from simple bag-of-words models like Tf-Idf or BM25 to more sophisticated topic models like

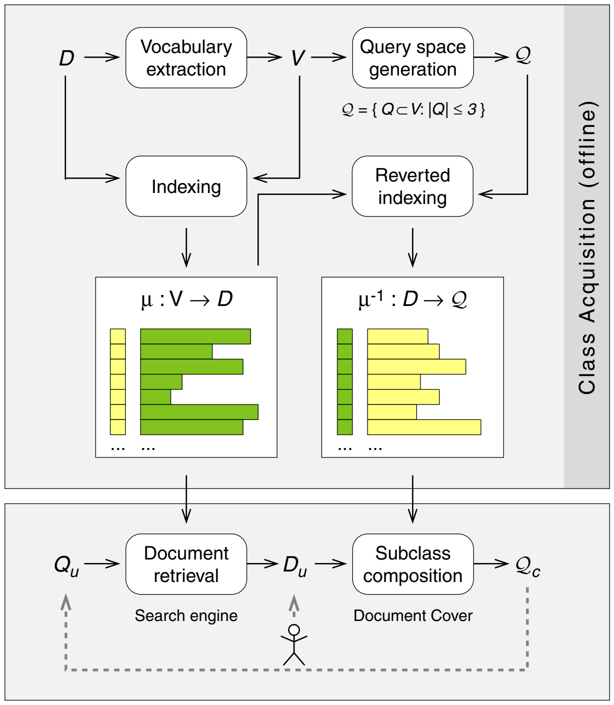

Figure 2: Strategy overview: How to for compose dynamic taxonomies for arbitrary library subsets. The upper part of the figure shows the offline component that continuously determines all keyqueries for the library documents and stores them in the reverted index. The lower part of the figure shows the online component that iteratively composes taxonomy levels $\mathcal{Q}_c$ for a user provided library subset D_u. The composition of sublevels is triggered by the user who selects a query Q_u from the current taxonomy level $\mathcal{Q}_c$.

LDA (latent topics) or ESA (explicit topics). Note that the choice of the retrieval model is crucial to our approach, and has a large impact on the quality of the composed taxonomies.

To acquire a set of classes for taxonomy composition from the search engine, we propose to generate a suitable subset $\mathcal{Q}$ of all queries that can be formulated from V, submit these queries to the search engine, and tag the returned documents with the respective query as class (right column of Figure 2). An efficient access to the classes (or queries) for a document is provided by a reverted index μ^{-1}. In the reverted index, each library document serves as the key for a postlist that contains all queries for which the document is relevant [20].

The queries that can be formulated on the basis of V theoretically comprise all $2^{|V|}$ subsets of V. To prune the query set, we consider only those queries as classes that meet the common perception of class labels in conventional classification systems. Specifically, we introduce the following two class label constraints:

1. *Part-of-speech constraint.* The parts of speech used for class labels in most classification systems are noun phrases (e.g., "information system" or "topic model"). We hence add to $\mathcal{Q}$ only queries that are combinations of noun-phrases.

2. *Query combination constraint.* A class label is usually made up of one noun phrase or two noun phrases joined by a conjunction (e.g., "digital libraries and archives"). In extreme cases, three phrases are joined. We hence discard from $\mathcal{Q}$ all queries that consist of more than three noun phrases.

Given the class label constraints, an upper bound on the number of queries can be stated as $|\mathcal{Q}| < \sum_{k=1}^{3} \binom{|V|}{k}$. Note that the actual number of queries will be lower, since the fraction of k-phrase queries that return a relevant document decreases for larger k. Following Stein et al. [11], we view the query space as the hypercube of phrase combinations illustrated in Figure 3 (the partitioning of the query space in the figure will be explained later). Each level l of the hypercube contains all l-combinations of phrases from V. To populate the reverted index efficiently, we explore the query space in a level-wise manner using the Apriori algorithm [1] starting with all single-phrase queries. For each subsequent level l, we consider only those l-phrase queries that can be generated by adding a phrase from V to an $(l-1)$-phrase query that returns more than one result. This way, the query space is pruned whenever a query reaches the maximum degree of specificity.

In addition to the class label constraints, we introduce a class assignment constraint controlling the fraction of a query's search results that are tagged with the query in the reverted index. The class assignment constraint ensures that the queries in a document's postlist are all keyqueries for the document (i.e., that the document is returned in the top results):

3. *Relevance constraint.* Users expect that documents assigned to a class in a taxonomy contain content *about* the subject represented by the class label, and do not just mention the class label in the text. However, determining at which position in a query result list the mere occurrence of the query phrases turns into "aboutness" is not trivial. Most retrieval models assign retrieval scores greater than zero already if a document contains any of the query phrases. To facilitate a binary decision upon the membership of a document to a class, we suggest the introduction of a retrieval score threshold: The retrieval score must exceed the score of a fictitious document with average length and average click rate that contains all query terms once.

The use of a retrieval score threshold may appear a bit old-fashioned compared to the use of click through information from the search engine's query log. But we argue that our constraint would account for users' click behavior if the employed search engine takes click behavior into account for retrieval score computation. Furthermore, a score threshold circumvents the cold-start problem inherent to any pure query log-based method (a query log is empty at first).

Once the postlists of the reverted index are filled with the library documents' keyqueries, the online process can compose dynamic taxonomies for user-provided library subsets. Before we turn to the online process, however, we emphasize that the reverted index is not a static data structure. Whenever the search engine indexes a new document, the reverted index allocates a new postlist for the document and computes the respective keyqueries. If the new document adds new noun phrases to V, the query space $\mathcal{Q}$ is extended accordingly, and the search results for the new queries are incorporated into the reverted index. Through this tight interplay with the search engine, our taxonomy composition approach always reflects the current state of the library.

3.2 Taxonomy Composition

Whenever a user turns to the system with a set of library documents D_u, the taxonomy composition process responds with a set of queries $\mathcal{Q}_c$ that represents the first level of the taxonomy for D_u. If the user selects a query from $\mathcal{Q}_c$ for further subdivision, the same process is repeated for that query's result set as the new D_u. This

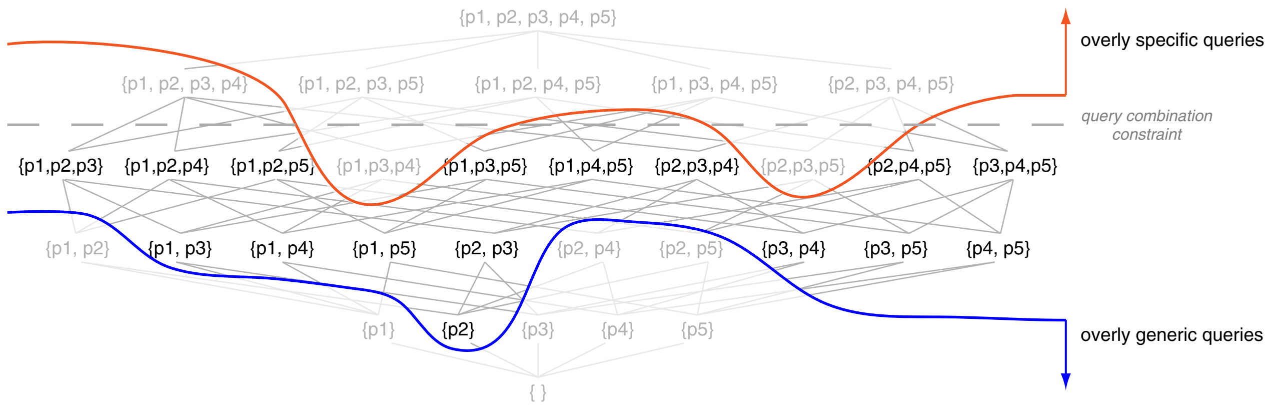

Figure 3: The search space for a five-phrase vocabulary $P = \{p_1, \ldots, p_5\}$. The two boundaries shown divide the search space into three subspaces of queries returning too few, the desired number, or too many documents (from top to bottom). The dashed line illustrates the query combination constraint prohibiting the use of more than three phrases in a query.

gives rise to an iterative process, where the taxonomy composition adapts interactively to the user's information need.

In each iteration, we strive for a subdivision into gradually more specific complementation classes which cover all the documents D_u and contain further topically similar library documents. To reach these goals, we introduce the following class composition constraints:

4. *Fan-Out Constraint.* In conventional classification systems, the maximum number k of subclasses into which a class can fall is about ten. For example, every internal class in the Dewey Decimal Classification System is divided into exactly ten subclasses. For our intended presentation of the composed taxonomy in a grid on a computer screen, we argue that twelve subclasses constitutes a more appropriate choice, and in addition allots more space for serendipity classes. Twelve subclasses can be placed evenly in a grid of one, two, three, four, six, and twelve columns, and hence allows a space efficient presentation on many devices from mobile to large desktop screens. Note that for the same reason, today's most popular responsive design framework, Bootstrap, uses a twelve-column grid.[5]

5. *Level of Generality Constraint.* One of the design goals for hierarchical classification systems is to compose a balanced taxonomy where each level constitutes a gradual specification of the parent classes [25]. Given a maximum fan-out of k, a balanced subdivision distributes the documents D of a parent class into disjunct subclasses of size $t = |D|/k$. Since the division into subclasses must be backed up by a semantic justification, the target subclass size t is only a rough reference point even in conventional classification systems. We thus add a margin m to t, and consider as candidates for the subclasses of a parent class all queries in $\mathcal{Q}$ with a search result size in the interval $[t - m, t + m]$. As a default, we suggest a margin of 50% of the target size.

The above level of generality constraint reduces the set of candidate queries that can be taken as subclasses in each iteration of the composition process. In Figure 3, the boundaries drawn into the query space indicate the separation of valid candidate queries from overly specific and overly generic ones.

[5]`http://getbootstrap.com`

To obtain the optimal set of subclasses in each iteration, we employ *Greedy Document Cover* given in Algorithm 1. Using the reverted index, the algorithm first compiles the set of candidate queries $\mathcal{Q}_u$. Each query from $\mathcal{Q}_u$ returns at least one of the documents from D_u, and satisfies the level of generality constraint.

The algorithm then finds an approximate solution for the optimization problem stated in Section 1.3 using a greedy set cover strategy [30]. It initializes the sets D_c (the document cover) and $\mathcal{Q}_c$ (the queries representing the subclasses for D_u) to the empty set. For up to k iterations, it then selects the query Q^* from $\mathcal{Q}_u$ that covers the maximum number of documents not yet covered by previous queries, and adds it to the subclasses $\mathcal{Q}_c$. This process is repeated until the maximum fan-out is reached, or no more candidate queries are available.

The queries in $\mathcal{Q}_c$, when submitted to the search engine, return a family of subsets of D_u. These can be displayed to the user as the classification of her search results. Note that, depending on D_u (and the corresponding queries in the reverted index), it may not be possible to construct a cover of the entire set D_u. In this case, the

Algorithm 1 Greedy Document Cover

```
Input:  reverted index $\mu^{-1}$, document set $D_u$,
        number of subclasses $k$,
        target class size $t$, margin $m$
Output: taxonomy sublevel $\mathcal{Q}_c$

   // Collect candidate queries for $D_u$ :
1: $\mathcal{Q}_u \leftarrow \bigcup_{d \in D_u} \{ Q \mid Q \in \mu^{-1}(d),\ |D_Q| \in [t \pm m] \}$
   // Build a document cover $D_c$ for $D_u$:
2: $\mathcal{Q}_c \leftarrow \emptyset$
3: $D_c \leftarrow \emptyset$
4: while $|\mathcal{Q}_c| < k \ \wedge\ \mathcal{Q}_u \neq \emptyset$ do
5:     $Q^* \leftarrow \operatorname{argmax}_{Q \in \mathcal{Q}_u} \{|D_Q \cap (D_u \setminus D_c)|\}$
6:     $\mathcal{Q}_u \leftarrow \mathcal{Q}_u \setminus \{Q^*\}$
7:     $\mathcal{Q}_c \leftarrow \mathcal{Q}_c \cup Q^*$
8:     $D_c \leftarrow D_c \cup D_Q$
9: return $\mathcal{Q}_c$
```

remaining documents in $D_u \setminus D_c$ may be added to a "miscellaneous" class. As previously noted, the algorithm is intended to be applied iteratively to subsets of D_u, including the query result sets for the classes in $\mathcal{Q}_c$. Since any label in the taxonomy is a valid query, retrieving the result set of a class for further partitioning is a simple matter of submitting the class label as a query to the search engine (illustrated through the arrow at the bottom of Figure 2).

3.3 Discussion

The constraints on candidate queries ensure that the set cover algorithm produces a reasonably well-balanced cover; at any level of the taxonomy, a given class can at worst be twice as large as any other. The reverted index must contain a sufficient number of queries satisfying the level of generality constraint for the documents in D_u. Otherwise, the dynamic classification cannot cover all of D_u, and will contain a large "miscellaneous" class.

Conversely, if the reverted index contains several disjoint queries near the upper bound of the required level of generality, the dynamic taxonomy may cover all of D_u in as few as $\frac{2}{3}k$ iterations. In this case, additional queries can be added to $\mathcal{Q}_c$ to form the *serendipity classes* discussed in Section 1. As a first basic approach, we pick serendipity classes randomly from the remaining candidate queries in $\mathcal{Q}_u$.

For the online part of our system, it is prudent to consider the runtime complexity since the response time of the digital library retrieval system needs to be kept within reasonable limits for the sake of user experience. Compiling the set $\mathcal{Q}_u$ of candidate queries requires $O(|D_u|)$ accesses to the reverted index, and at most $O(|\mathcal{Q}_u|)$ operations to select from the results those queries that fit the required specificity bounds. In order to solve the actual document covering, the algorithm requires $O(|\mathcal{Q}_u|)$ queries to the search engine (independently of the number of iterations), as well as $O(|\mathcal{Q}_u|)$ set-intersection operations per iteration.

Considering that this procedure is repeated iteratively for all subclasses in the dynamic taxonomy, the amount of computation may seem prohibitive for guaranteeing responsiveness at query time. However, only the initial document covering of D_u needs to be computed at query time. While the user studies the first taxonomy level, the computation of subsequent taxonomy levels can be initiated and the respective subclasses be cached. This introduces a tradeoff between storage and processing requirements at preprocessing versus at query time, and middle-ground solutions are conceivable. For instance, analysis of the query logs for the digital library's search engine can motivate long-term caching for frequently used taxonomy classes, while rarely issued classes are computed online.

In this respect, taxonomy composition can be framed as an instance of a *slow search* problem, a class of search scenarios where traditional speed requirements are relaxed in favor of a high quality search experience [26]. Due to the top-down nature of the document cover algorithm, the taxonomy composition follows the user's exploration of the information need, generating class subdivisions on demand.

4. EMPIRICAL EVALUATION

Due to the combinatorial complexity of the search space, dynamic taxonomy composition is clearly a big data problem. Even for relatively small collections, the size of the query space quickly grows to a magnitude that requires the use of distributed processing techniques in order to be at all feasible. For the experiments reported in this section, we employ the MapReduce programming model [9], as implemented by the Apache Hadoop software library.[6]

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

Our experiments run on a 40-node cluster of off-the-shelf desktop machines, each equipped with four 2.4GHz CPU cores and 8–16 GB of RAM.

In order to evaluate the utility and feasibility of our proposed dynamic classification system, we apply our approach to a dataset of 30,000 scientific papers—manually collected from the ACM digital library over the course of several years—which we refer to as Webis-CSP-Corpus. The corpus contains various metadata, including conference, year of publication, and the classification according to the ACM CCS taxonomy. With our experiments, we explore the following research questions:

1. What additional processing requirements are imposed by the dynamic classification system, both online and offline?

2. How useful are dynamic taxonomies in fulfilling a profiling-oriented information need?

3. How well do dynamic taxonomies support serendipity search or document recommendation? How do they compare to human-made taxonomies in this regard?

As a preprocessing step, we extract a set of keyphrases from each document in the collection, using the approach proposed by Barker et al. [4], in which prominent head noun phrases function as keyphrases for a document. We use the log-linear part-of-speech tagger implementation by Toutanova et al. [28] to identify candidate noun phrases. For the purpose of our experiments, the extracted keyphrases form the vocabulary V which gives rise to the queries in the search space. While we do not aim to replace traditional static classification systems entirely, they serve as baseline for the experiments described below.

4.1 Processing requirements

To map out the processing requirements for dynamic taxonomy composition, we generate the query space on corpus subsets of various sizes, and then compute a document cover for the set of all documents in the collection. For each year from 1990 through 2010, we take from the Webis-CSP-Corpus those documents published that year or earlier. Table 1 shows the number of documents, the number of keyphrases in the vocabulary and the resulting query space sizes for a sample of years, as well as the corresponding offline and online processing times. Note that the figures reported for the size of the query space $\mathcal{Q}$ refer to the pruned search space shown in Figure 3—the set of all possible queries constructed from the vocabulary is larger by several orders of magnitude.

As discussed in Section 3, offline processing involves acquiring a set of candidate queries from which to compose taxonomy classes. We implement class acquisition as a processing pipeline of MapReduce jobs which evaluate the Okapi BM25 retrieval function. By computing document relevance scores for the entire collection in parallel, this approach facilitates the large number of query evaluations needed to compile $\mathcal{Q}$. If our system were implemented in a digital library setting, this type of batch processing could easily be integrated with the library's other data preprocessing efforts. As the penultimate column of Table 1 shows, even using our modestly-sized Hadoop cluster, the investment in processing time is manageable, and much smaller than the time required for a taxonomy built by human experts.

Digital library users would expect their search results to be classified and displayed in a timely fashion. This constrains the time available to compute the document cover for the initial query result set D_u. Since starting up a MapReduce job involves significant overhead, the document cover for user queries needs to be handled

Table 1: Query space sizes and processing times for corpus subsets of different magnitudes.

Year	Dataset sizes			Run time [h:m:s]	
	$\|D\|$	$\|V\|$	$\|\mathcal{Q}\|$	$\mathcal{Q}$	Cover
1990	4,784	3,670	174,605	6:17	4.9 s
1995	8,593	5,982	607,615	14:39	7.1 s
2000	12,901	8,431	1,677,217	35:21	11.3 s
2005	19,288	11,893	4,577,178	1:24:24	17.7 s
2010	28,950	15,807	13,516,167	3:09:00	27.2 s

differently. For this part of the experiment, we use an implementation of the document cover algorithm running on a single machine. The two indexes required for the computation are accessed via an efficient inverted index implementation developed in-house. The numbers in the last column of Table 1 report the time needed to compute the document cover for the entire collection on a single 2.4GHz CPU core. The 27 seconds of processing time on the largest subset of the collection is arguably too long to guarantee a responsive user experience. However, the experiment represents a worst case, in that the document cover is computed for the entire collection. Whereas in the main use case for dynamic classification we envision—classification of search result sets—the set of documents to cover is much smaller. This impacts performance greatly, as the last column of Table 1 indicates. Each additional document to cover implies an access to the reverted index, yielding additional queries, each of which implies an additional access to the inverted index. In the reverted index for Webis-CSP-Corpus, each individual document is associated with an average of 10,000 queries, which supports the intuition that smaller subsets of the collection will be significantly less costly to process.

4.2 Profiling

A user with a profiling-oriented information need (e.g., identifying all the classes of the papers from a specific conference) is best served by a classification system that subdivides the document set in a maximally informative way. We investigate a dynamic classification system generated for the documents in the Webis-CSP-Corpus, and compare it to the static, human-made ACM CCS classification system. Owed to its organic growth over time, we expect the static taxonomy to exhibit a lower information content than the dynamic classification system.

The ACM CCS taxonomy for the documents in the Webis-CSP-Corpus is three levels deep. The median number of subdivisions under a given CCS class is 11, 6 and 3, for the top, middle, and leaf level of the taxonomy. For the sake of comparability, we generate a dynamic taxonomy of the same depth, with the fan-out parameters set to the CCS median at each level. As we are comparing taxonomies for the entire corpus, the document set D_u input to the covering algorithm comprises the entire collection D.

Figure 4 shows the distribution of taxonomy class sizes for both the human-made and the dynamic classification system. The left chart shows taxonomy classes taken from the corpus documents' metadata. On the right, the classification system was generated by the document cover algorithm, using queries formulated from keyphrases extracted from the documents' text. In each chart, bar height shows the total number of documents covered per depth of the classification. Both classification systems assign more than one class label to some documents—hence the number of documents covered by the classification exceeds the total number of documents in the collection. The shading of the bars illustrates how class size is distributed among the branches of the taxonomy.

The figure demonstrates the high level of imbalance in the human-

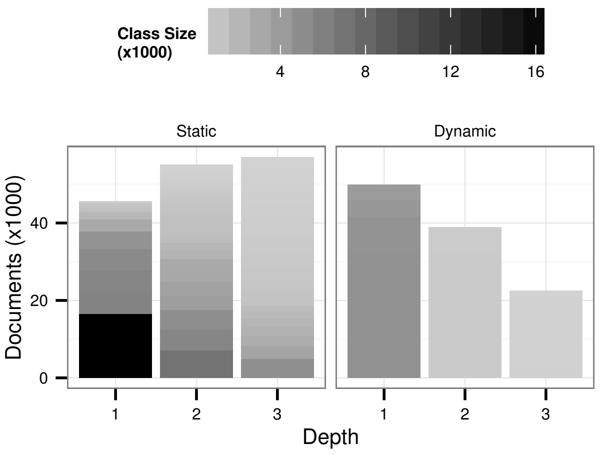

Figure 4: Distribution of class sizes in the human-made taxonomy, and a dynamic classification system generated from extracted keyphrases.

made taxonomy—especially on the highest level of classification. Here, the largest branch subsumes more than 16,000 documents, whereas any other class is at most one third that size. By contrast, the dynamic query-based classification system results in a taxonomy where the number of documents per class is uniform within any given level, and distinct across levels. Owed to the class size imbalance, the static taxonomy covers a much larger number of documents on the middle and leaf level. For instance, the static taxonomy contains a leaf class of comparable size to the top-level classes in the dynamic classification system.

To assess the usefulness of each classification system in a profiling-oriented task, we compute the entropy at each branch in the classification tree, as well as the number of redundant document-class assignments. Table 2 shows the results aggregated by hierarchy level. With entropy, we measure the information content in bits of the class assignments, averaged over each level of classification. Entropy is measured at each branch in the taxonomy and corresponds to the probability of a randomly sampled document belonging to a given subclass below the branch—it is maximal if all subclasses are the same size. The dynamic taxonomy achieves higher entropy than the human-made classification system on the upper two levels of the hierarchy, whereas the human-made classification system carries more information at the leaf level. Both taxonomies are similar with respect to the number of classes per document. While outliers in the dynamic classification system tend to have higher redundancy than in the static one, the average document is assigned to two classes at

Table 2: Comparison of entropy and redundancy for dynamic and human-made classification systems in the Webis-CSP-Corpus by hierarchy level.

Depth	Avg. Entropy	Classes per document				
		μ	σ	Min	Median	Max
Static						
1	2.79	1.59	0.74	1	1	5
2	1.94	1.93	1.01	1	2	8
3	1.84	2.11	1.16	1	2	10
Dynamic						
1	3.46	1.93	1.01	1	2	7
2	2.58	2.19	1.47	1	2	13
3	1.58	1.93	1.32	1	1	19

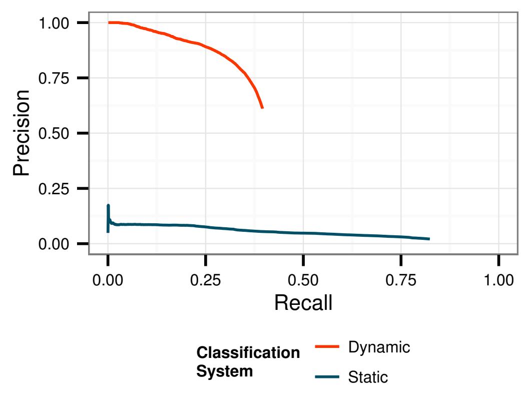

Figure 5: Average precision-recall curve, showing the performance of the dynamic and human-made classification systems at the document-recommendation task.

each level of classification in both cases.

These results demonstrate that it is possible to generate a dynamic taxonomy with desirable structural properties, using only automatically extracted head noun phrases as class labels. The dynamic taxonomy outperforms the human-made reference taxonomy especially with respect to the information content of the higher-level subdivisions of the document set. However, the human-made taxonomy is superior in other regards, and we expect the best performance using a retrieval model that combines information from both sources—our described idea of backing up a traditional classification with a dynamic query-based component.

4.3 Document Recommendation

In a final avenue of inquiry, we investigate the suitability of dynamic classification systems for document recommendation. In this scenario, the user has already retrieved a set of documents relevant to her information need. The retrieval system's task is to present related relevant documents, helping the user uncover new aspects of the subject under investigation. Dynamic classification systems, being tailored to the user's specific information need, should perform considerably better at this task than a static classification system that has to take the entire document collection into account.

To test this hypothesis, we perform the following experiment: First, we sample a random query Q from the reverted index. This query represents the overall information need. Second, we randomly partition the result set of Q into two subsets D^+ and D^- of equal size, where D^+ represents the known aspect of the information need, and D^- represents the unknown aspect (i.e., the set of documents that the retrieval system should recommend). Third, we construct a dynamic classification system for D^+ using queries from $\mathcal{Q} \setminus Q$. Fourth, we measure the number of documents from D^- that are retrievable via the leaf classes of the classification system.

Table 3: Distribution of maximum recall, and precision at maximum recall, for the human-made and the dynamic taxonomy in the document recommendation task.

	μ	σ	Min	Median	Max
Static					
Max. Recall	0.82	0.04	0.76	0.84	0.89
Max. Precision	0.02	0.01	0.01	0.02	0.05
Dynamic					
Max. Recall	0.40	0.11	0.26	0.34	0.62
Max. Precision	0.61	0.08	0.47	0.63	0.73

We model a user who scans all leaf classes for relevant documents. For the purpose of this experiment, the hypothetical user pursues an optimal strategy: in each step, she examines the class that contains the maximum number of previously unseen documents from D^-. The user stops when all retrievable documents from D^- have been considered. For each taxonomy class the user considers, we measure the cumulative recall for hidden relevant documents, that is, the proportion of documents from D^- that the user has retrieved so far. We also measure the precision with respect to D^- at each step. In total, we perform forty runs of the above experiment, with different initial information needs. To put the performance of the dynamic classification system into context, we perform the same experiment using the leaf classes of the static CCS taxonomy, using the same optimal search strategy as above.

Figure 5 shows a set of precision-recall curves depicting the performance averaged over all forty experiments, comparing the static and dynamic classification system. As the hypothetical user examines taxonomy classes, each curve plots the fraction of all documents in D^- that the user has seen (recall) against the fraction of previously unseen documents from D^- out of all retrieved documents (precision). The curves end at the point of maximum recall, where no further documents from D^- can be found in the taxonomy. Due to the much larger leaf classes, the human-made taxonomy achieves better maximum recall, at the cost of much lower precision. In the dynamic taxonomy, nearly all documents are relevant to the user's information need, and out of these, close to half are related to the unknown aspect of the information need. Table 3 shows the distribution of maximum recall for D^- and precision at maximum recall, over all forty experiments. In other words, the numbers in the table show the distribution for the rightmost point in the precision recall curve for the hidden relevant documents. The table demonstrates a larger trend: while the large leaf classes in the static taxonomy allow for consistently higher recall, the ratio of relevant documents found is much better for the dynamic classification system.

5. CONCLUSIONS AND FUTURE WORK

The rise of digital content in libraries has a deep impact on the foundations of library management and forces us to revisit and rethink the established library tools and processes. In this paper, we revisit library classification systems and analyze their application portfolio in the digital era. What we find is that classification systems are no longer urgently required to retrieve relevant documents, but are rather supposed to complement the abilities of a query-based search engine. Classification systems are a valuable tool for digital libraries if they provide structured access to any given library subset, if they complement the set with further relevant documents, and if they embed the relevant documents into a broader context. To this end, we present a dynamic taxonomy composition framework, which acquires its classes directly from keyqueries against the library's search engine. This close connection to the search engine brings a variety of advantages compared to existing approaches, including cost-efficient maintenance and seamless integration into search interfaces. Our experiments demonstrate the feasibility of our approach despite the complexity of the query space.

For future work, we propose a qualitative assessment of our framework under different retrieval models. Most promising in this regard seems the use of explicit topic models in combination with lexical knowledge bases. We expect a retrieval model of this kind to improve the quality of the class labels compared to standard bag of word models. This is especially likely for macroscopic concepts that appear only rarely as words in the documents' text.

6. REFERENCES

[1] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proc. 20th Int. Conf. Very Large Data Bases*, San Francisco, CA, USA, 1994, pp. 487–499.

[2] R. Baeza-Yates et al. Query Recommendation Using Query Logs in Search Engines. In *Proc. 2004 Int. Conf. Current Trends in Database Technology*, Berlin, Heidelberg, 2004, pp. 588–596.

[3] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval - The Concepts and Technology Behind Search, 2nd edition*. Pearson Education Ltd., Harlow, England, 2011.

[4] K. Barker and N. Cornacchia. Using Noun Phrase Heads to Extract Document Keyphrases. In *Advances in Artificial Intelligence, 13th Biennial Conf. Canadian Society for Computational Studies of Intelligence*, Montréal, Quebec, Canada, 2000, pp. 40–52.

[5] F. Bonchi et al. Topical Query Decomposition. In *Proc. 14th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, New York, NY, USA, 2008, pp. 52–60.

[6] C. Carpineto et al. A Survey of Web Clustering Engines. *ACM Comput. Surv.*, vol. 41, July 2009, pp. 1–38.

[7] S.-L. Chuang and L.-F. Chien. Towards Automatic Generation of Query Taxonomy: A Hierarchical Query Clustering Approach. In *Proc. 2002 IEEE Int. Conf. Data Mining*, Washington, DC, USA, 2002, pp. 75–82.

[8] D. Cutting et. al. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. In *Proc. 15th Annu. Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Copenhagen, Denmark, 1992, pp. 318–329.

[9] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, vol. 51, 2008, pp. 107–113.

[10] E. Gabrilovich and S. Markovitch. Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. In *Proc. 20th Int. Joint Conf. Artificial Intelligence*, Hyderabad, India, 2007, pp. 1606–1611.

[11] T. Gollub et al. From Keywords to Keyqueries: Content Descriptors for the Web. In *Proc. 36th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Dublin, Ireland, 2013, pp. 981–984.

[12] G. Henry. *Core Infrastructure Considerations for Large Digital Libraries*. Council on Library and Information Resources, Digital Library Federation, 2012.

[13] H. Hu et al. Diversifying Query Suggestions by Using Topics from Wikipedia. In *Proc. 2013 IEEE/WIC/ACM Int. Joint Conf. Web Intelligence and Intelligent Agent Technologies*, Atlanta, GA, USA, 2013, pp. 139–146.

[14] D. Koller and M. Sahami. Hierarchically Classifying Documents Using Very Few Words. In *Proc. 14th Int. Conf. Machine Learning*, San Francisco, CA, USA, 1997, pp. 170–178.

[15] R. Li et al. DQR: A Probabilistic Approach to Diversified Query Recommendation. In *Proc. 21st ACM Int. Conf. Information and Knowledge Management*, New York, NY, USA, 2012, pp. 16–25.

[16] X. Liu et al. Automatic Taxonomy Construction from Keywords. In *Proc. 18th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, New York, NY, USA, 2012, pp. 1433–1441.

[17] H. Ma et al. Diversifying Query Suggestion Results. In *Proc. 24th AAAI Conf. Artificial Intelligence*, Atlanta, GA, USA, 2010, pp. 1399–1404.

[18] C. Manning et al. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[19] R. Navigli et al. A Graph-based Algorithm for Inducing Lexical Taxonomies from Scratch. In *Proc. 22nd Int. Joint Conf. Artificial Intelligence*, Barcelona, Spain, 2011, pp. 1872–1877.

[20] J. Pickens et al. Reverted Indexing for Feedback and Expansion. In *Proc. 19th ACM Conf. Information and Knowledge Management* Toronto, ON, Canada, 2010, pp. 1049–1058.

[21] H. Poon and P. Domingos. Unsupervised Ontology Induction from Text. In *Proc. 48th Annu. Meeting Assoc. for Computational Linguistics*, Stroudsburg, PA, USA, 2010, pp. 296–305.

[22] S. Robertson et al. Simple BM25 Extension to Multiple Weighted Fields. In *Proc. 13th ACM Int. Conf. Information and Knowledge Management*, New York, NY, USA, 2004, pp. 42–49.

[23] Y. Song et al. Post-ranking Query Suggestion by Diversifying Search Results. In *Proc. 34th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, New York, NY, USA, 2011, pp. 815–824.

[24] B. Stein et al. Beyond Precision@10: Clustering the Long Tail of Web Search Results. In *20th ACM Int. Conf. Information and Knowledge Management*, Glasgow, UK, 2011, pp. 2141–2144.

[25] A. Taylor and D. Joudrey. *The Organization of Information*. Library and Information Science Text Series. Libraries Unlimited, Inc., 2009.

[26] J. Teevan et al. Slow search: Information Retrieval Without Time Constraints. In *Proc. Symp. Human-Computer Interaction and Information Retrieval*, New York, NY, USA, 2013, pp. 1:1–1:10.

[27] E. Toms. Serendipitous information retrieval. In *Proc. 1st DELOS Network of Excellence Workshop Information Seeking, Searching and Querying in Digital Libraries*, Zurich, Switzerland, 2000, pp. 11–12.

[28] K. Toutanova et al. Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network. In *Proc. 2003 Conf. North Amer. Chapter Assoc. for Computational Linguistics and Human Language Technology*, Stroudsburg, PA, USA, 2003, pp. 173–180.

[29] S. Tuarob et al. Automatic Tag Recommendation for Metadata Annotation Using Probabilistic Topic Modeling. In *Proc. 13th ACM/IEEE-CS Joint Conf. Digital Libraries*, New York, NY, USA, 2013, pp. 239–248.

[30] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[31] D. Weinberger. *Everything Is Miscellaneous: The Power of the New Digital Disorder*. Henry Holt and Company, 2007.

[32] R. White and R. Roth. *Exploratory Search: Beyond the Query-Response Paradigm*. Synthesis Lectures on Information Concepts, Retrieval, and Services Series. Morgan & Claypool, 2009.

[33] Y. Yang et al. Query by Document. In *Proc. 2nd ACM Int. Conf. Web Search and Data Mining*, New York, NY, USA, 2009, pp. 34–43.