An Online Analytical System for Multi-Tagged Document Collections

by

Grzegorz Drzadzewski

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2015

Abstract

The New York Times Annotated Corpus and the ACM Digital Library are two prototypical examples of document collections in which each document is tagged with keywords and significant phrases. Such collections can be viewed as high-dimensional document cubes against which browsers and search systems can be applied in a manner similar to online analytical processing against data cubes. The tagging patterns in these collections are examined and a generative tagging model is developed that can mimic the tag assignments observed in those collections. When a user browses the collection by means of a Boolean query over tags, the result is a subset of documents that can be summarized by a centroid derived from their document term vectors. A partial materialization strategy is developed to provide efficient storage and access to centroids for such document subsets. A customized local term vocabulary storage approach is incorporated into the partial materialization to ensure that rich and relevant term vocabulary is available for representing centroids while maintaining a low storage footprint. By adopting this strategy, summary measures dependent on centroids (including bursty terms, or larger sets of indicative documents) can be efficiently and accurately computed for important subsets of documents. The proposed design is evaluated on the two collections along with PubMed (a held-back document collection) and several synthetic collections to validate that it outperforms alternative storage strategies.

Finally, an enhanced faceted browsing system is developed to support users' exploration of large multi-tagged document collections. It provides summary measures of document result sets at each step of navigation through a set of indicative terms and diverse set of documents, as well as information scent that helps to guide users' exploration. These summaries are derived from pre-materialized views that allow for quick calculation of centroids for various result sets. The utility and efficiency of the system is demonstrated on the New York Times Annotated Corpus.

Acknowledgements

I would like to thank my supervisor, Frank Wm. Tompa, for all his guidance, valuable insight, constructive feedback, and a lot of patience, as I went through my research journey over the years. Thanks to him, I was introduced to interesting areas of research, became a better researcher, and eventually, completed this thesis. I also thank Evangelos Milios, Lukasz Golab, and Wayne Oldford for giving me their time and helping me during various stages of my research. In addition, I would like to thank Jimmy Lin and Luis Gravano for taking the time to review and improve this work. I am also thankful to my parents and Doris Cheung, for their encouragement to pursue my passions and their support throughout my many years of study.

Preliminary versions of the work presented in this thesis have appeared in the following publications:

- Grzegorz Drzadzewski and Frank Wm. Tompa. Exploring and analyzing documents with OLAP. *In Proceedings of the 5th Ph.D. Workshop on Information and Knowledge Management (PIKM)*, 2012. (Introduction to the problem)
- Grzegorz Drzadzewski and Frank Wm. Tompa. Partial materialization for online analytical processing over multi-tagged document collections. *Knowledge and Information Systems (KAIS)*, 2015. (Chapters 3 and 5)
- Grzegorz Drzadzewski and Frank Wm. Tompa. Enhancing exploration with a faceted browser through summarization. *In Proceedings of the 15th ACM SIGWEB International Symposium on Document Engineering (DocEng)*, 2015. (Chapter 7)

I would like to thank the anonymous reviewers of those papers along with the reviewers of previous submissions for their constructive feedback.

This research was supported by the University of Waterloo, NSERC's PGS program, NSERC's Business Intelligence Network, NSERC's Discovery Grant program, Mitacs, and the Ontario Graduate Scholarship program.

Dedication

To Doris, my favourite distraction…

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Large document collections such as the New York Times Annotated Corpus (the NYT) and the ACM Digital Library (ACM) cover many diverse topics. It can be a daunting task to decide what to read when approaching a new topic or to find which combinations of topics deserve more attention. As an aid to readers, various tags (usually keywords and significant phrases) are assigned to each document in these collections, reflecting the topics covered by that document. In a large collection, each tag can be assigned to hundreds or thousands of documents.

Following standard practice in information retrieval, we model a document as a *bag of terms* represented by a *document term vector* (DTV), which is a vector of values where each entry corresponds to a term together with the term's (normalized) frequency in that document. A set of DTVs can be aggregated to obtain a *set centroid* for the corresponding documents, which can be used to summarize the document set. Given the centroid, a system can produce other summary measures, such as a representative set of "bursty" terms [Goorha and Ungar, 2010, Popescul and Ungar, 2000], the medoid [Gelbukh et al., 2003, Cutting et al., 1992], or a diverse set of indicative documents [Deolalikar, 2014].

Access to a collection of documents with rich metadata can be facilitated with the use of *facets* [Teevan et al., 2008], where facets represent dimensions or aspects of a subject according to which documents in a collection are organized. Common facets include topical categories, location, colour, time, etc. Each facet consists of a set of values that are organized in a flat or hierarchical structure. A document can have multiple facets and zero or more values assigned from each facet. A collection of documents with facets can be navigated by using facet values as filters. For a set of selected facet values, a faceted browser will show all documents that satisfy the specified filters. Facets can be combined with keyword search to produce what is referred to as *faceted search*. Faceted search enhances traditional search by providing users with additional information about the result set, which includes some information about the facets and facet values found in the result set. It also allows the user to iteratively select from the provided facet values to refine the query and narrow down the search results.

In addition to the DTV, each document in our collections is assigned a *set of tags* that are external to the document. Each metadata tag is a value chosen from a facet that corresponds to a conceptual dimension used to describe the data [Yee et al., 2003]. For simplicity, we assume that the facets are unstructured (i.e., that the value space within a facet is not hierarchically organized) and that each document is assigned zero or more values from each of the facets. Following standard practice, we assume that the assigned tags have been selected with care: They are typically of high quality and identify topics or important concepts found in the document.

A facet-based browsing environment supplements a traditional search engine by adding facilities that allow users to benefit from the metadata tags. With the help of faceted search a user may start to explore the ACM tagged document collection by issuing a traditional search request, say `[databases cloud]`. As in other systems, the user is presented with the top k matching documents, but in addition the user is also informed by the system of the tags associated with those documents. In response to `[databases cloud]`, the user might learn that all the corresponding documents are tagged “database,” 90% are tagged “cloud computing,” 55% of the top responses are also tagged “service-oriented architecture,” 35% are tagged “security and privacy,” and 10% are tagged “genome informatics.” (Instead of precise percentages, similar information might instead be provided in the form of tag clouds.) The user could then select tags of interest to formulate a refined query by issuing a Boolean query over tags (i.e., “slicing and dicing” the collection).

Faceted search helps to narrow a large result set down to a more manageable size for browsing, and a study at the University of North Carolina library showed that it was preferred by users over traditional search interfaces based on text content alone [Ramdeen and Hemminger, 2012]. In a variety of other settings, user studies have found that systems supporting faceted search and browsing are superior to traditional search engines and to systems based on clustering by document content [Fagan, 2013]. For example, Yee et al. [Yee et al., 2003] found that “Despite the fact that the interface was often an order of magnitude slower than a standard baseline, it was strongly preferred by most study participants. These results indicate that a category-based approach is a successful way to provide access to image collections.” Kipp and Campbell [Kipp and Campbell, 2010] found that “Users would find direct access to the thesaurus or list of subject headings showing articles indexed with these terms to be a distinct asset in search.” Hearst [Hearst, 2006] concluded that "Usability results show that users do not like disorderly groupings like [those produced by clustering systems], preferring understandable hierarchies in which categories are presented at uniform levels of granularity." Pratt et al. [Pratt et al., 1999] found that "a tool that dynamically categorizes search results into a hierarchical organization by using knowledge of important kinds of queries and a model of the domain terminology ... helps users find answers to those important types of questions more quickly and easily than when they use a relevance-ranking system or a clustering system." Zhang and Marchionini [Zhang and Marchionini, 2005] found that "[A faceted search and browsing] interface will bring users added values beyond simple searching and browsing by in fact combining these search strategies seamlessly." Faceted search has thus emerged as a valuable technique for information access in many e-commerce sites, including Wal-Mart, Home Depot, eBay, and Amazon [Tunkelang, 2009]. Development of faceted search interfaces has also been the focus for companies such as Endeca[1], whose interfaces are used by many websites.

We envision an enhanced interface that, in addition to a traditional faceted search interface, provides summary information about the resulting document set. As the first step in this direction, we want to provide summary information about document sets resulting from queries issued over tags. For example, a summary may consist of the k most representative articles in the sub-collection that satisfies the query, the most common terms used within articles in that sub-collection, and the distribution of tags that are assigned to articles in the sub-collection. If the summary matches the user's information need, individual articles in that set can be retrieved; otherwise the user can reformulate the query (often narrowing down the result set by specifying additional tags or expanding it by removing some tags from the query) to arrive at a more appropriate set of articles. For an analyst (or even a casual reader) armed with the New York Times, this approach might uncover sets of articles that provide a comprehensive summary of news reports on a specific subtopic of interest. For a computer scientist investigating an unfamiliar research area through the ACM Digital Library, providing summaries based on tag-based queries can identify the most relevant articles to read in the area and how those articles relate to topics identified by other tags.

We rely on the previous studies to validate the utility of faceted search and browsing: It is prevalent, effective, and satisfies users' needs. In this dissertation, we concentrate on making such systems more efficient. If a document collection is already provided with meaningful metadata tags so that faceted search and browsing is feasible, the main problem that needs to be solved is to find a fast way of calculating centroids, which are required to provide summaries of document sets that match users' tag-based queries. Because some

[1]http://www.endeca.com (now part of Oracle)

sets can be very large, aggregating large amounts of data in order to calculate summary measures may be too time-consuming to be performed online.

For conventional data warehouses, *Online Analytical Processing* (*OLAP*) systems have been developed in order to speed up the aggregation of multidimensional data through full or partial materialization of summaries. In these systems, data cubes serve as the model for describing OLAP operations [Gray et al., 1997]. A cubes' dimensions reflect the attributes that characterize the facts stored in the cube's cells; for example, a set of sales records might have dimensions for date of sale, location of sale, type of product sold, customer demographics, etc. Thus, OLAP dimensions correspond to facets, and attribute values correspond to facet values. However, this direct mapping between facets and OLAP dimensions only exists when each document in the collection has at most one facet value that comes from each facet [Ben-Yitzhak et al., 2008]. When documents have multiple values that come from the same facet, the facet values need to be mapped to the multidimensional schema (Section 2.6.1) or single-dimensional schema (Section 2.6.2) in order for them to be modeled with a data cube. For example, if we had a document that gives a review for a romantic comedy film, then that document would have two values from the genre facet assigned to it: romance and comedy. Rather than storing the genre facet as a dimension of an OLAP cube, its facet values need to be mapped to either a multidimensional or a single-dimensional schema. We adapt OLAP's approach to partial materialization in order to provide summaries at each step of a faceted search when space is limited.

Unfortunately, current OLAP systems are designed for data collections that have tens of dimensions and will not work for document collections that have hundreds of facets with millions of tags. To handle such a large number of dimensions, we propose to materialize centroids for sub-collections that correspond to all documents sharing small subsets of tags. Thus, centroids are stored for predetermined subsets of the data, and calculating centroids for arbitrary subsets corresponding to users' queries requires aggregating data from several overlapping subsets (because documents with multiple tags will contribute to multiple materialized centroids). The techniques used in current OLAP systems, however, do not accommodate such overlap.

An additional problem that needs to be addressed involves the choice of vocabulary used for representing the stored centroids. Due to the extremely large number of unique terms found in document collections (the NYT has a vocabulary size of approximately 2 million unique terms), storing a full shared global vocabulary that is used for representing centroids of all classes would consume an unacceptable amount of space. As a result, it is feasible to store only a small subset of this global vocabulary. However, the vocabulary available for representing a centroid of any given class has a direct effect on the summary information that the user will receive. If the vocabulary contains only a few terms that are informative for the given class, then the indicative terms and documents that are derived from its centroid may not provide meaningful summaries. As a result, the popular approach of using a reduced shared global vocabulary that is formed by taking the top k informative terms from each class may not be effective in enabling the creation of informative class summaries. Instead an alternative local vocabulary storage approach needs to be developed that provides a rich vocabulary for representing class summaries and has a low storage cost. The large vocabulary found in both the NYT and ACM document collections makes this an important problem that needs to be addressed.

This dissertation is focused on proving the following statement:

Thesis Statement: Efficient calculation of text centric aggregate measures on sets of documents defined through queries over tags in multi-tagged document collections can be accomplished by deriving the measures from materialized views. There exists a partial materialization strategy that can simultaneously satisfy the following four requirements:

1. allows fast computation of aggregate measures,
2. provides a rich vocabulary,
3. consumes a small amount of storage, and
4. supports document collections with a wide variety of realistic tagging patterns.

1.1 System Requirements

In this section, we describe requirements and associated challenges for a system that will support online analytical processing for a large document collection. The requirements are derived in part by examining characteristics of the PubMed interface to biomedical literature [Ebbert et al., 2003].

PubMed includes more than 24 million abstracts and corresponding citations to articles, which are annotated with a variety of tags[2] chosen from Medical Subject Headings (MeSH), the National Library of Medicine controlled vocabulary thesaurus used for indexing articles

[2]For consistency within this dissertation, we continue to use "tag" to refer to a metadata term assigned to an article from a controlled vocabulary, even though PubMed's use of "tag" refers to the attachment of a facet label to a query term.

for PubMed; the EC/RN Number, assigned by the Food and Drug Administration (FDA) Substance Registration System for Unique Ingredient Identifiers; and Supplementary Concept tags, which include chemical, protocol or disease terms. The PubMed interface supports searching for documents using a standard text search, matching query terms against the abstract, the citation, and all assigned metadata tags, as well as by specifying that some of the query terms should be restricted to matching MeSH terms (or some other facet) only.

PubMed users looking for relevant articles can benefit immensely from searching with the aid of metadata tags [Mosa and Yoo, 2013]. PubMed is a very large collection and the sizes of sets of search results are often large. Because the search results are returned in reverse chronological order (unlike results returned by web search engines that are ordered by presumed relevance to the query), its users can certainly benefit from more efficient calculation of aggregate measures that summarize the contents of query results.

1.1.1 Supported Measures

As explained earlier, a document is considered to be a bag of terms and is represented by its document term vector (DTV). To conserve space and avoid storing stop words and other uninformative terms, only m terms are stored in each DTV; their corresponding frequency values are normalized by the length of the document. The available choices for choosing the m terms to store as part of each document's DTV are described in Chapter 6. For the time being we assume that there is a shared global vocabulary and the same m terms are stored for each DTV in the collection.

In order to provide meaningful summaries about a document set (e.g., set of diverse documents indicative of the contained topics, set of indicative terms, or many other multi-document summaries), we need to compute the set centroid C, which can be represented by a vector of term frequencies equal to the mean of all the DTVs for documents that belong to that set. However, instead of storing the means directly, for a set of documents S, we store its centroid C_S as a dictionary that maps terms to *sum* values along with the count of documents found in the set, which is then easily updated when documents are added to or removed from the set:

$$C_S[term].sum = \sum_{d \in S} d[term] \tag{1.1}$$

$$C_S.size = |S| \tag{1.2}$$

where d is a normalized DTV of length m. Thus, a set centroid vector has length m regardless of how many documents are in the set. From the stored measures, the mean frequency of a term in document set S is derived using Equation 1.3.

$$C_S[term].freq = \frac{C_S[term].sum}{C_S.size} \tag{1.3}$$

In addition to *sum*, each term in S stores the count of documents in the set that have the corresponding term:

$$C_S[term].count = |\{d \in S \mid d[term] > 0\}| \tag{1.4}$$

The stored set size along with the counts of documents with each term are used to calculate the *mutual information* of terms, which is required for feature selection (Section 2.4.6).

1.1.2 Supported Queries

Associated with each document is a set of "metadata" tags, each of which is assumed to represent some aspects of the document's content. We allow the user to pose queries as Boolean formulas over tags, such as $Election \wedge President \wedge (Stocks \vee Stock\_Market)$. Conjunctions of terms narrow down the scope of documents to those that involve all the concepts represented by the conjuncts. The use of negation is allowed, but only in the form "and not" to allow a conjunction with the complement of the documents having a given term, as in the example $President \wedge \neg Election$. Disjunction provides a means of query expansion, allowing synonyms and related tags to be included in a query [Manning et al., 2008]. After an initial query, a user might issue subsequent queries that are refinements of the original query and include one or more additional conjuncts (slicing/dicing). This type of behaviour can occur when using a browsing interface to navigate a collection, as is explained in the next section.

1.1.3 Expected Workload

Users explore a multi-tagged document collection through a browser front end that enables them to invoke Boolean queries. As part of their exploration, they may pose queries and read summaries (in the form of data derived from set centroids, such as a set of diverse documents indicative of the contained topics or a set of indicative terms). After users examine summaries of document sets, they may choose to narrow down the result set by

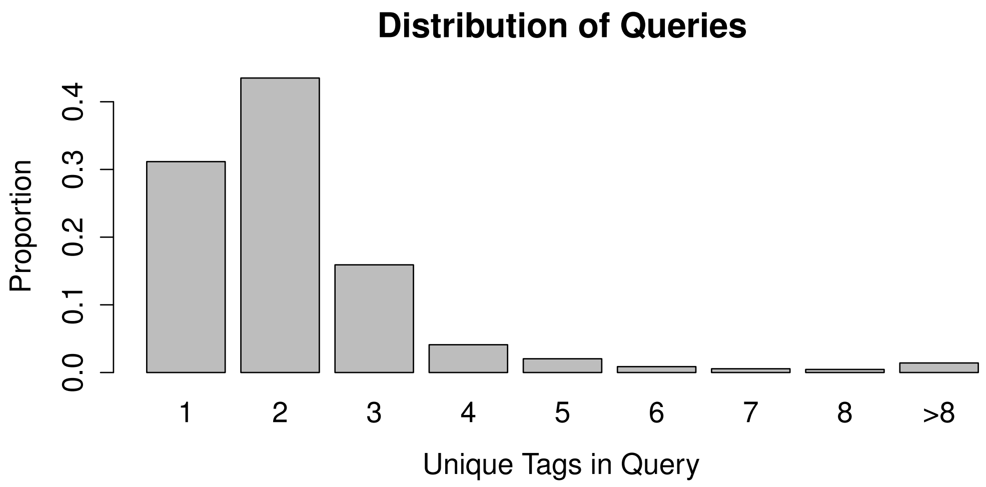


Figure 1.1: Analysis of unique tags per query

issuing more specific queries. The browsing system we are developing is required to provide quick responses to the generated queries.

We rely on data from a PubMed query log[3] [Mosa and Yoo, 2013] to help characterize a feasible query workload. Among 2,996,301 queries collected over a single day, 16,928 queries include only terms chosen from facets that have a controlled vocabulary (specifically, MeSH terms [MH], MeSH major topics [MAJR], MeSH subheadings [SH], filters [FILTER], EC/RN Numbers [RN], and supplementary concepts [NM]), with the possible addition of one or more pure text terms. Treating each text conjunct or disjunct as if it were a single tag, these queries involve anywhere from 1 to 46 tags, with the majority of the queries using between 1 and 3 tags (Figure 1.1).

Figure 1.2 tallies the number of distinct tags (y-axis) that have been found to co-occur with k other tags (x-axis), as observed in the PubMed query logs. For example, assume that the query log consists of the following four queries: Music, Music AND Jazz, Music AND Opera, Music AND Movies OR Theatres. In that query log, the tag 'Music' appears in queries with four other tags and is included in the tally corresponding to the bar at $x = 4$. The tags 'Movies' and 'Theatres' each co-occur with two tags, and the tags 'Jazz' and 'Opera' co-occur with only one tag. The query log analysis provided in Figure 1.2 shows that in the queries posed by users, some tags appear together with many different

[3]Available at ftp://ftp.ncbi.nlm.nih.gov/pub/wilbur/DAYSLOG.

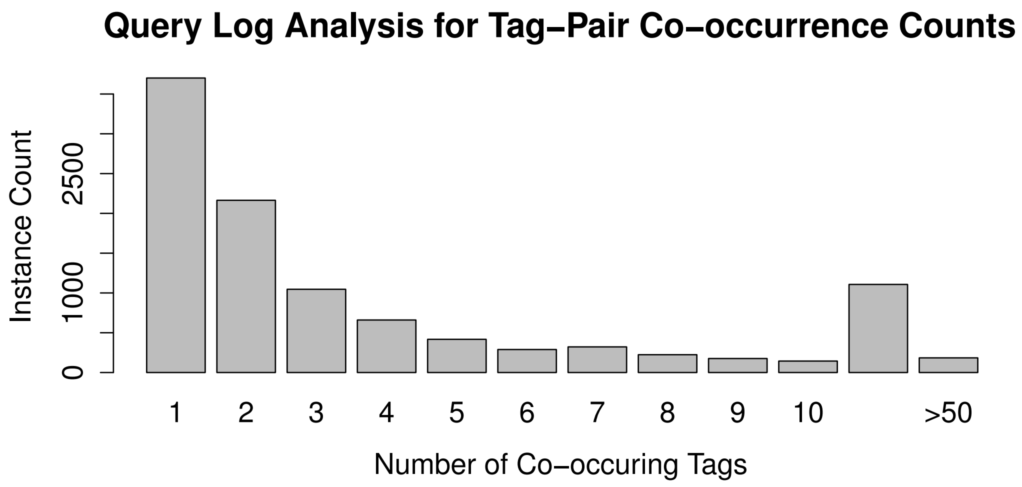


Figure 1.2: Number of distinct tags co-occurring with tags in a query

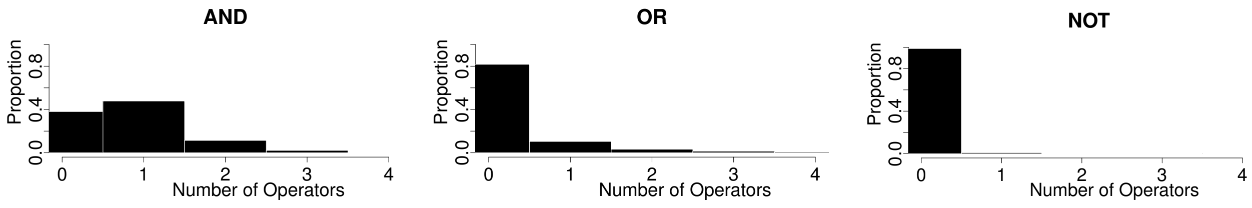


Figure 1.3: Distribution of operator counts in PubMed query log: a) AND, b) OR, c) NOT

tags.

The usage patterns of the three Boolean operators are summarized in Figure 1.3, which shows that the "NOT" operator occurs in 1% of queries, the "OR" operator occurs in 18% of the queries, and the "AND" operator occurs in 62% of the queries; 31% of queries are of length 1 and so use no operators. These observations suggest that a realistic query workload will likely include primarily short queries that are predominantly conjunctive, as has been observed for other Web search systems before the inclusion of auto-completion for suggesting long queries [Jansen et al., 2000].

In summary, our expectation is that (after some preliminary traditional searches of the text) users explore a collection by starting with a small set of tags of interest and then iteratively refining their queries to be more focused by including additional tags. For some queries, query expansion will be applied to incorporate some alternative tags. Thus, most queries will be conjunctions of tags (i.e., no negations and only occasionally disjunctions to accommodate alternative tags), most queries will be short, and most queries will match sets that include a large number of documents.

1.1.4 Query Processing Model

We assume that documents are stored as files and that the collection is indexed by a mapping from document IDs to the corresponding files. In order to support queries over tags, we further assume the existence of an inverted index that stores a postings list of document IDs for each of the tags. Finally, we assume that normalized document term vectors have been precomputed and that an index from document IDs to DTVs (as might be produced by a standard search engine) is also available.

With this minimalistic storage structure, tag queries may be answered using the following steps:

1. Use the inverted index over tags to return the set S of document IDs that satisfy the query.
2. Initialize the document set centroid C_S to be empty.
3. For each $s \in S$:
   (a) Retrieve DTV for s.
   (b) Add DTV to document set centroid C_S using Eqs. 1.1 and 1.4.

This algorithm requires $|S|$ DTVs to be read from secondary storage, which will be quite slow for large sets. Even for systems with sufficient main memory to store the whole collection, it will be beneficial to avoid online aggregation of $|S|$ documents, especially when supporting many concurrent users. Therefore, it is desirable to bound the number of documents that must be retrieved for each query.

One way to reduce the cost of answering a query is to store precomputed centroids for well-chosen sets of documents $\mathbb{P}$ and to use these at query time to reduce the number of documents that must be retrieved. Given a document set S, we find a highly overlapping set $P \in \mathbb{P}$ and retrieve the precomputed set centroid C_P as well as the DTVs for all documents in $S \triangle P = (S - P) \cup (P - S)$, the symmetric difference between sets S and P. To calculate C_S, the DTVs of the retrieved documents are added to or subtracted from the centroid of P (Eqs. 1.1 and 1.4) in order to compensate for the difference between S and P. As a result, instead of retrieving $|S|$ DTVs, we need to retrieve $|S \cup P| - |S \cap P|$ DTVs as well as C_P. As a special case, $S \in \mathbb{P}$, in which case we merely need to retrieve C_S and avoid accessing any DTVs.

When choosing $\mathbb{P}$, the sets for which we precompute and store centroids, its size is constrained by the amount of available storage space. To benefit from $\mathbb{P}$, we wish to select sets that closely match the expected query load. To this end, we adopt the practice of using partial materialization of document cubes [Harinarayan et al., 1996].

1.1.5 Design Objective

We wish to provide a fast response to user queries by having an upper bound on the number of documents or centroids of materialized sets that need to be retrieved from secondary storage. At the same time we wish to minimize the number of set centroids that need to be precomputed and materialized to accomplish this. We focus on providing upper bound guarantees on execution costs for (positive) conjunctive queries, since they are expected to be most frequent: For each such query, no more than k DTVs or set centroids need to be accessed, for some fixed k. Queries that involve disjunction and negation will be answered using multiple conjunctive subqueries, and they may therefore require more than k DTVs or set centroids in total.

A document cube provides an excellent mechanism for structuring the collections of documents so as to answer Boolean queries on tags. Each cell in the cube represents the set of documents that have a specific tag assignment, and each cuboid represents document sets that are aggregated ("rolled up") by grouping on specific tags and ignoring others. As a result, conjunctive tag queries can be answered by selecting specific cells from appropriate cuboids, and centroids of document sets that correspond to other Boolean queries can be computed by combining the centroids from selected cuboid cells. The problem to be addressed is to determine which cells or cuboids to materialize to balance space and time. In addition, for each materialized cell, we wish to ensure that the stored vocabulary is large and relevant for the queries that it will be used to answer, so that the indicative terms and indicative documents generated from the centroid are of high quality while consuming minimal amount of space.

1.2 Contributions

This thesis includes the following contributions:

- Detailed analyses of tagging patterns in two representative multi-tagged document collections: the New York Times Annotated Corpus and the ACM Digital Library;
- development of a generative tagging model for the tagging patterns observed in the NYT and ACM collections;
- a storage design that permits fast calculation of centroids for document sets that result from both short and long conjunctive queries over tags, and supports aggregation of overlapping sets;
- development and evaluation of several partial materialization strategies for high-dimensional, sparse data;
- development and evaluation of several vocabulary storage techniques;
- application of the proposed partial materialization strategy to support various text centric document set summary measures; and
- implementation of an enhanced faceted browser that takes advantage of the developed partial materialization to efficiently provide the user with summaries for various sets of documents that the user is navigating through.

1.3 Thesis Overview

The thesis is organized as follows. Background and related work on folksonomies, tag recommendation systems, browsing systems, facet selection, document set summarization, OLAP for data warehouses, document warehouses, and topic modeling are summarized in Chapter 2. The properties of two prototypical multi-tag document collections are examined in Chapter 3. Based on the observed properties of the two collections, a generative tagging model is proposed and fitted to the two collections in Chapter 4; this generative model is used to generate additional synthetic data. A novel individual cell materialization strategy that supports efficient computation of document set centroids and a set of partial materialization techniques built on top of it are described and evaluated in Chapter 5. The evaluation is conducted on the NYT, ACM, PubMed and several synthetic collections generated from the derived generative tagging model. Various vocabulary storage strategies are explored and evaluated in Chapter 6. An enhanced faceted browsing system that supports efficient access to summary operations over sets of documents is developed in Chapter 7. Finally, conclusions and future work are summarized in Chapter 8.

Chapter 2

Background and Related Work

2.1 Folksonomies and Tag Recommendation Systems

The New York Times and the ACM Digital Library rely on tags being chosen by users with care so as to maximize the reuse of tags where applicable and to distinguish between concepts through the use of disjoint tag sets where possible. To achieve these ends, the collections employ controlled vocabulary for some facets and allow only limited use of uncontrolled vocabulary.

In contrast, many social Web sites, such as Delicious and Flickr, allow users to attach arbitrary tags to documents to organize content on the Web. Tags can be chosen by users at will, and different users may assign different tags to the same object. This results in so-called folksonomies [Hearst, 2009] that include many tags per document, large tag vocabularies, and significant noise. Faceted browsing has been implemented over folksonomies in systems such as dogear [Millen et al., 2006], and the complex tagging patterns involved can benefit from more efficient exploration, which is the aim of our work.

To help reduce noise, there is much research on tag recommender systems, which are designed to help users assign tags to documents. We expect that good recommender systems would approximate users' tagging behaviour in order to provide the correct tags. Also, as tag recommender systems become more popular, their use will influence users' tagging behaviour, in which case we can expect that the tags suggested by a good tag recommender system will correspond to the tags assigned to a document.

User studies that examine users' perceptions of the role and value of tags [Kim and Rieh, 2011] showed that one common view of tags is as keywords (tags describe key aspects of the document) and another common view is for categorization. This is supported by another study [Ames and Naaman, 2007], a taxonomy of tagging motivations in ZoneTag/Flickr, which concluded that one of the purposes of tags is to show context and provide content description.

Content-based tag recommender systems assume that tags for a specific resource can be extracted by processing the textual information about the resource to be annotated. These approaches adopt techniques from information retrieval [Baeza-Yates and Ribeiro-Neto, 1999] in order to extract relevant terms to label a resource. More specifically, term frequency and inverse document frequency have been shown to yield good keyword identification results [Budzik and Hammond, 1999, Efthimiadis, 1995, Witten et al., 1999], and their use has been adopted by tag recommendation systems [Brooks and Montanez, 2006]. Content authors and editors do not explicitly compute inverse document frequency when tagging an article, but their intuition regarding which words are informative replaces the use of this measure by human judgment.

In related work, a tag recommender system that relies on topic modeling was developed to provide an annotator with a set of diverse tags that represent the various topics covered in the document [Bi and Cho, 2013]. The generative model in the system simulated the users' tagging process in a social tagging system. It assumed that for any resource there are a multitude of topics, and that when users tag a resource, they first identify topics of interest from the resource, after which they express the chosen topics via a set of words (tags). Each topic accordingly corresponds to a probability distribution over tags, which gives the probability of picking out a tag with respect to a certain topic. The user studies performed as part of the evaluation of the system suggested that users preferred the tags suggested by this new system. In another tag recommender system, a model based on Latent Dirichlet Allocation (LDA) was also shown to perform well [Krestel et al., 2009].

From these previous studies, we conclude that a carefully annotated document has tags representing all the topics that have sufficiently high presence in that document.

2.2 Browsing Document Collections

In place of faceted search, which has been described in the introductory chapter, browsing systems might rely on document clustering. For example, traditional search engines are designed to provide a user with a ranked list of documents that satisfy a query, and clustering may be performed on top of the result set in order to organize similar documents into groups [Zamir and Etzioni, 1999]. Because clustering can be fully automated, it can be applied to text collections that have not been assigned metadata tags. However, if clustering is applied online and if it were to be applied to result sets consisting of thousands of documents, it would impose unacceptably long delays. To avoid this bottleneck, systems such as Clusty.com perform clustering on the top k results only. On the other hand, if clustering is applied offline, cluster labels can be interpreted to be metadata tags and the techniques proposed here can be similarly applied.

Scatter/Gather is a well-known document search interface based on clustering [Cutting et al., 1992]. Users explore a document collection by dynamically clustering a set of documents (scattering), selecting clusters of interest based on their summaries, and then treating all documents in the selected clusters as one set (gathering). These steps are then repeated to further investigate the contents of the sub-collection. The summaries used to characterize clusters take the form of a set of representative terms, chosen on the basis of frequency alone, together with the headlines of the documents closest to the centroids.

Like Scatter/Gather, our proposed system allows users to repeatedly select subsets of the document collection, examining summaries for each grouping of documents to determine whether or not to include specific groupings in the refinements. However, unlike Scatter/Gather, the system we envision is based on a multi-valued, faceted labeling for each document rather than on hard clusterings; thus even if cluster labels at each step were to be treated as if they were metadata tags for externally-specified classes, Scatter/Gather would correspond to a single-valued labeling of documents. Furthermore, in Scatter/Gather it is difficult for users to predict what clusters will be generated since the grouping criterion is hidden, unlike when aggregation is specified through tags visible to user. Finally, we envision a search system in which a user is free to broaden the search at any step, rather than being expected to restrain themselves only to narrowing down the result set.

To make Scatter/Gather usable in an interactive manner, offline hierarchical clustering can be performed on the document collection [Cutting et al., 1993]. In this approach, meta-documents corresponding to a union of documents are created offline, and during the scatter phase the meta-documents are clustered instead of the actual documents, thus reducing the number of items to be clustered and thereby reducing execution time. Document clustering is therefore only approximated. In addition, inter-document distances are computed based on selected features instead of the full text of the meta-documents, thus again reducing execution time. Interestingly, the third variant of our proposal stores centroids for meta-classes, somewhat akin to Scatter/Gather's meta-documents, but those centroids are corrected to exact centroids for the associated classes before they are used in browsing.

An even faster implementation of Scatter/Gather (LAIR2) was developed by Ke et al. [Ke et al., 2009], based on precomputing a complete (binary) hierarchical clustering of the documents. Thus, for a collection with N documents, LAIR2 materializes $N - 1$ clusters. Then, instead of clustering documents during the browsing stage, it retrieves prematerialized nodes from the cluster hierarchy. Like the previous approach, however, the authors are only concerned with improving the execution time and do not consider the storage cost required to store every sub-cluster of a full hierarchical clustering. In contrast, the amount of storage required by a browsing system, as well as execution time, is central to our work.

One difficulty in browsing via tags is to determine which tags are present in the collection and how tags are related to each other. A query and browsing interface can display the distribution of tags that are assigned to articles in each result set, thereby suggesting tags that can be used for further refinement. For broadening a search, the system could display tags that are associated with carefully chosen supersets of the result set. Alternatively, the system could provide a mechanism to browse the tags themselves (as opposed to the documents associated with those tags) through an interface to a thesaurus or ontology [Côté et al., 2010, Mu et al., 2014, Shiri et al., 2011]. We make no assumptions about the structure of the tag space for our work, and the incorporation of tag-browsing facilities is orthogonal to our work.

2.3 Facet Selection

As was mentioned in the introductory chapter, facets and facet values (corresponding to tags) presented to the user by the faceted browser play an important part in guiding users' exploration of the result set. The challenge with presenting facets and corresponding facet values, especially for large result sets, is that there often are too many available facet values, which leads to information overload for the users [Tunkelang, 2009]. To address this issue, a variety of facet and facet value selection algorithms have been proposed that reduce the number of facet values shown [Kashyap et al., 2010, Li et al., 2010, Roy et al., 2008, Dash et al., 2008]. Facet value selection is driven by the goal of minimizing the navigational cost. Our approach towards producing an indicative set of documents relies on having access to a well chosen set of tags that are made available by facet selection algorithms.

Before the faceted values can be selected, we first need to identify all the candidates and their corresponding counts for all documents in a result set. This can be expensive to compute online for large result sets. One approach to improving the computation speed of the document counts has been proposed by Dash et al. [Dash et al., 2008]. The approach relies on compressed bitmaps for storing the posting lists of documents in an inverted index for each tag and a novel bitset tree directory structure that reduces the number of unnecessary bitset intersections that are performed between tags.

2.4 Document Set Summarization

When navigating through many document sets it may often be too time consuming to examine all documents found in each document set of interest. For this reason document sets are often summarized. By viewing a set summary, a user can get a quick overview of its contents. Common measures for summarizing sets of documents include cluster centroid, cluster labels, representative documents, and multi-document summarization. Efficient access to the centroids of corresponding classes will be beneficial to most implementations of these measures.

2.4.1 Cluster Centroid

A cluster centroid has been used in various works as a measure for summarizing the contents of a group of documents. In Scatter/Gather [Cutting et al., 1992] a cluster profile (which is equivalent to a centroid) was used for summarizing groups of documents. This cluster profile was further used to derive other group summaries such as the most frequent terms in the profile. In the MiTexCube system [Zhang et al., 2011], sets of documents have been summarized by first clustering them and then representing each cluster by their centroid.

2.4.2 Cluster Labeling

Standard clustering algorithms group documents, after which cluster labeling needs to be performed on the generated clusters in order to produce descriptive, human-readable labels. One approach towards generating cluster labels is through the use of a differential cluster labeling technique, which picks terms by comparing the distribution of terms in one cluster with that of another [Manning et al., 2008]. One popular differential cluster labeling technique picks terms that are frequent and predictive [Popescul and Ungar, 2000] as defined in Equation 2.1.

$$p(term|cluster) \times \frac{p(term|cluster)}{p(term)} \tag{2.1}$$

A concept of 'bursty' terms has been used in temporal text analytics. This is similar to differential cluster labels, in that it identifies terms that experience higher frequency of occurrence for some segment of time as compared to their usual frequency of occurrence. Goorha et al. used the definition given in Equation 2.2 to extract bursty terms in a Twitter feed [Goorha and Ungar, 2010] by comparing a term's usage in some chosen class of documents of interest against that term's usage in the corpus as a whole. A variant of the bursty term measure has also been used in analyzing blog posts [Bansal and Koudas, 2007].

$$\frac{(term\_count\_in\_class)}{(term\_count\_in\_corpus)^{0.95}} \tag{2.2}$$

In both the cluster labeling and bursty terms identification scenarios, the best candidate terms are frequent in a collection and have higher relative frequency in the subset than in the superset.

Since document clustering is usually performed with few features, it is important to note that cluster labels need to be derived from the full text of the documents and not from the subset of features used in clustering. Thus, one needs to evaluate the relative frequencies of each term found in centroid vectors of multiple clusters. Having quick access to all the required centroids can allow for fast generation of labels.

2.4.3 Indicative Documents

Another popular approach to summarizing a cluster is through a representative sample of documents. A representative sample of a large collection (where the sample has a similar word distribution as the population from which it is taken) has the number of documents coming from each topic proportional to its size. Such a sample has multiple documents coming from a dominant topic, and smaller topics may not be included at all. This approach may not provide an appropriate level of diversity in the representative set, and, as a result, it might limit the amount of new information a user gets from reading each additional document. Alternatively, a diverse set of documents that provides a broad perspective on the different topics found in the cluster might be more appropriate. Users studies showed that most users prefer to see a diversified sets of documents for representing clusters in aggregator websites [Abbassi et al., 2013].

A common approach to representing a cluster is through a medoid, which is a document that has the lowest average cosine distance to all other documents in the cluster [Gelbukh et al., 2003] or equivalently has the lowest cosine distance to the centroid. A representative set of size k consists of k documents closest to the centroid. The Scatter/Gather algorithm [Cutting et al., 1993] uses the titles of such documents to summarize the generated clusters. However, it turns out that medoids often do not make good "representative" documents because the large number of dimensions found in the cluster can cause what is called the "curse of dimensionality" [Rajaraman and Ullman, 2011], where due to the large number of dimensions, all documents are equally far from one another and, in turn, far from the centroid. Under such circumstances it was shown by Deolalikar [Deolalikar, 2014] that documents that "cover" the concepts represented in the centroid, picked with a coverage-based approach, make significantly and consistently better representative documents than the documents that are closest to the centroid. A similar approach of picking documents as representatives of a cluster based on their coverage of terms found in the cluster centroid was also previously described in [Gelbukh et al., 2003].

Whether the closest to centroid or "cover" based approach is used for picking representative documents, in both cases, access to the centroid of the document set is required.

2.4.4 Search Result Diversification

There is a large body of research on search result diversification [Santos et al., 2015]. However, that work deals with diversification of results to ambiguous text queries, for which different results are optimal depending on the interpretation of the query. That body of work is not directly applicable to our scenario since we do not deal with queries that are issued over text, but instead the queries are issued over tags that are assumed to be well-chosen and far less ambiguous.

Search queries over text produce a ranked list of results based on the tf-idf measure, where the top k ordered results are derived efficiently from the inverted index and the diversification is performed on the ordered list afterwards. For example, the Maximal Marginal Relevance (MMR) [Carbonell and Goldstein, 1998] search result diversification approach is initialized with a ranked lists of results (ordered by query relevance), from which the top ranked document is selected as the first representative. Each following document is iteratively selected, such that the query relevance and distance from previously selected documents are maximized. This selection approach ensures that all the chosen documents are relevant to the query and have low redundancy.

Search queries over tags produce a result set of documents, where there is no ordering. In our situation, documents are assigned single instances of tags (we are not considering folksonomies, where the same tag can be assigned to a document by many users). As a result, documents matching a query over tags cannot be ordered based on the tag frequency.

Ordering can be imposed on the documents in the result set by using the text found in the documents. If we have access to the centroid of the result set, we can order the documents in the result set by their cosine distance to the centroid or by their coverage of the important terms found in the centroid [Deolalikar, 2014]. For a large result set, ordering the documents can be time consuming. An inverted index is not designed to support the calculation of documents' centroid distance, and so, it may be necessary to retrieve the DTVs of all documents in the result set and calculate the distances directly. However, once an ordering is imposed on the result set, search result diversification approaches can be applied.

2.4.5 Multi-document Summarization

Multi-document summarization techniques are used to generate a textual summary for a set of related documents. The generated summary is a shorter version of the source text which preserves the overall meaning and information content. The text summarization methods can be classified as extractive and abstractive. An extractive summarization method selects important sentences or paragraphs from the original document and concatenates them into a shorter form. The importance of each sentence is decided based on its statistical and linguistic features. Abstractive summarization does not reuse existing sentences or words, but instead generates sentences based on the statistical and linguistic features directly.

There are many different approaches to perform such multi-document summarization, based on abstraction and information fusion, topic-driven summarization, clustering, graphs, and ranking [Gupta and Lehal, 2010, Das and Martins, 2007]. Of particular relevance here are multi-document summarization methods that rely on using the centroids of document sets [Radev et al., 2004], which is the measure for which we are designing an efficient infrastructure.

2.4.6 Feature Selection

A major difficulty associated with text clustering or categorization is the high dimensionality of the feature space. The native feature space of a document collection consists of all the unique terms that occur in the collection. Even in a moderate-sized text collection, there can be tens or hundreds of thousands of unique terms [Yang and Pedersen, 1997]. Such a feature space is prohibitively large for many learning algorithms. As a result, it is highly desirable to reduce the size of the feature space without sacrificing the accuracy of the categorization or clustering algorithms. The size of the feature space also has an impact on the total storage cost of the collection, which is an important concern that needs to be addressed when designing a materialization strategy for a document warehouse. From the perspective of storage cost and clustering, it is desirable to have a small feature space, so that resulting clusters can be computed quickly and their summaries take little space. However, in order to handle nonprevalent topics, such that they can be detected, and documents belonging to them correctly categorized or clustered, the available feature space used to describe the collection has to be rich. As a result, the feature space needs to be large enough to support the rarer topics but small enough so that space is not wasted on uninformative features.

A conventional approach for reducing the feature space in natural language processing is to remove all stop words, as well as those that globally occur fewer than five times [Apté et al., 1994]. Language specific stop words, such as 'the' in English, occur very frequently and convey little information about the documents. The infrequent words are unreliable indicators for use as features and often are the result of misspelling. The feature space can be reduced further by normalizing all the terms (converting to lower case, stemming, removing punctuation, etc.) and removing numerical values from consideration.

In addition to the above text processing techniques, there exists a multitude of feature selection techniques that pick specific terms out of the original vocabulary to be used for clustering. These rely on document frequency thresholding, information gain, mutual information, χ^2 testing, or term strength [Yang and Pedersen, 1997, Garnes, 2009, Sebastiani, 2002]. This is in contrast to feature extraction techniques, such as latent semantic indexing, that transform the set of features available into a new and typically smaller set of features [Garnes, 2009, Sebastiani, 2002]. Since a feature produced by feature extraction techniques may not correspond to any one term, such features do not make good candidate labels for clusters.

We do not perform document clustering or classification in our work, but we do need to select vocabulary (the features) that will be used to characterize a set of documents. As part of our preprocessing we rely on *mutual information* $I(W;C)$ for performing feature selection [Manning et al., 2008], as defined in Equation 2.3

$$I(W;C) = \sum_{c \in C} \sum_{w \in W} p(w,c) \log \left(\frac{p(w,c)}{p(w)p(c)} \right) \tag{2.3}$$

where W corresponds to a random variable indicating the presence (absence) of a term in a document and C corresponds to a random variable indicating the presence (absence) of a tag assignment to a document, $p(w,c)$ is the joint probability of a specific term and tag being assigned to a document, $p(w)$ is the probability of a word occurring in a document, and $p(c)$ is the probability of a tag in question being assigned to document.

Feature selection methods produce a ranked list of features which it is hoped are good towards training a classifier for one specific category. That is, feature selection methods are usually specific to the category (set of documents) being characterized [Garnes, 2009]. Category-specific features are often called *local*. Because it is often impractical to have a separate set of features for each category, often the ranked feature lists are merged into one common feature list that is used for all categories. Such a category-independent list of features is called *global*. When there exist n categories, a common approach to merging the features includes taking the top k/n features from each category and creating a global feature list of size k. Another merging approach is to measure the average score of each feature over all n categories and then picking the top k features based on their average score [Manning et al., 2008].

In a comparative analysis between a local and global dictionary approach, Apté et al. showed that the local dictionary approach drastically reduces the dimensionality [Apté et al., 1994]. As the number of topics in a collection grows, a global dictionary with even as many as 10,000 words will be insufficient to handle rare topics, but increasing its size will further increase the dimensionality problem. The local dictionary approach was shown to be both faster and more accurate than relying on a global dictionary (which usually contains features that are less specialized for the categories). However, even with all these advantages of using a local feature set, the global approach is predominantly used because the benefit of having a common document representation usually outweighs the loss of accuracy [Manning et al., 2008].

2.5 OLAP for Data Warehouses

As was explained in Chapter 1, OLAP systems are used to aid users with multidimensional analysis. Because multidimensional analysis often requires aggregated measures over some of the dimensions (e.g., average sale prices for each product per day, regardless of location and customer), OLAP systems provide the materialization of selected *cuboids* defined over a subset of dimensions, storing precomputed aggregates in each resulting cell. The dimensionality of a cuboid is equal to the number of unaggregated dimensions, and the space is proportional to the number of cells (the product of the number of possible values in each unaggregated dimension). Thus a d-dimensional cuboid stores aggregated values in cells indexed by the possible values for each of the d unaggregated dimensions, and if each dimension is binary requires $O(2^d)$ space.

2.5.1 Full Materialization

OLAP systems that materialize all possible cuboids offer the best response time to user queries. However, full materialization requires $O(2^n)$ space for cubes with n dimensions. Compression can be applied to achieve full materialization while reducing the storage cost; this can save space in situations where there is significant repetition in cell measures, as is the case with sparse cubes. Compression techniques for data cubes include condensed cubes [Wang et al., 2002], dwarf cubes [Sismanis et al., 2002], and quotient cubes [Lakshmanan et al., 2002]. However, these techniques do not scale to a high number of dimensions [Li et al., 2004].

2.5.2 Partial Materialization

Partial materialization techniques are used to materialize a subset of cuboids (also referred to as views) from the lattice of cuboids [Harinarayan et al., 1996]. When answering a query, instead of fetching the data from the base cuboid and performing aggregation on it, the cuboid corresponding to the query can be calculated from the closest materialized superset cuboid. Therefore, the subset of cuboids to materialize is picked so as to minimize the time needed for the expected query workload, while requiring no more than a given amount of storage.

Thin cube shell materialization is a partial materialization where only the base cuboid and certain low-dimensional (most highly aggregated) cuboids are stored [Li et al., 2004]. More specifically, in addition to the base cuboid, the strategy stores all cuboids having exactly d dimensions, where $d \ll n$, n is the total number of dimensions, and there are $\binom{n}{d}$ d-dimensional cuboids. Alternatively, we could materialize all cuboids having d or fewer dimensions, which would further reduce the execution time of short queries at the expense of additional storage space. However, d-dimensional cuboids can be used to answer queries that involve at most d dimensions only; this involves choosing a materialized cuboid and aggregating the data for the dimensions omitted in the query. On the other hand, queries involving more than d dimensions are answered by aggregating over the base cuboid. Picking a larger d for materialization results in increased storage cost and increases the time required to calculate queries with few dimensions, but picking a small d results in much longer computation time for queries with more than d dimensions. If the expected workload has a wide range of queries, there may not be a fixed d that is appropriate.

As an improvement over a thin cube shell, Li et al. [Li et al., 2004] proposed a shell fragment approach for dealing with high-dimensional cubes. The technique relies on the assumption that high-dimensional data have limited interactions among dimensions (tags). It assumes that on average any one tag interacts with at most K other tags, where K is at most five and these tag interactions can usually be well clustered. Under such circumstances when a collection has T unique tags, it can be partitioned into T/K nonoverlapping fragments. Depending on the properties of the data and the query workload, it may be necessary to choose fragments of various sizes. However, larger fragments require more storage space. If the tag interactions cannot be clustered well, it may be necessary to store overlapping fragments to provide satisfactory query response time, in which case more fragments need to be stored. This, in turn, leads to greater storage requirements. For each of these fragments a full cube materialization is stored; thus, all the cuboids of dimensions ranging from 1 to K are materialized. This results in $2^K - 1$ cuboids materialized per fragment, where a cuboid with d dimensions has $2^d - 1$ cells, which therefore implies $\sum_{i=1}^{K} \binom{K}{i}(2^i - 1)$ cells for a fragment. For a fragment of size $K = 3$, 19 cells per fragment are needed. For scenarios in which the prematerialized fragments do not enclose the user's query, again the view needs to be calculated from the base cuboid, which can be time-consuming.

Recently, a 'Smart Cubes' [Antwi and Viktor, 2014] approach was developed as a further improvement over the shell fragment. Like the shell fragment approach, it partitions the dimensions into fragments, but to conserve space, it performs a partial materialization on each of the fragments. In addition, the smart cubes system is designed to adapt to change in query patterns. If it starts receiving new queries that require access to dimensions that span multiple fragments, it will dynamically replace old materialized fragments with new ones, so that it maintains a satisfactory response time for the new query workload.

2.6 Document Warehouses

A document warehouse is like a data warehouse, except that instead of performing analyses over tabular data, it supports analyses over documents. OLAP systems designed for document warehouses require dimensions that provide useful organization of documents and text centric measures that are appropriate for analyzing the text found in the documents. Dimensions for storing tags were developed to allow the analysis of tagged collections [Jin et al., 2010]. In other work, a textual dimension was created to enhance the functionality of OLAP on documents by providing a term hierarchy for the terms found in documents [Lin et al., 2008]. This term hierarchy allows the user to roll-up and drill-down on the terms between very specific and general terms. A topic dimension that consist of a topic hierarchy was developed to allow users to explore documents based on the topics they cover [Zhang et al., 2009, Mendoza et al., 2015]. A dimension that contains an ontology of keywords was integrated with another OLAP system [Inokuchi and Takeda, 2007]. OLAP in document warehouses has been used to provide users with summaries of related documents through the use of centroids of the clusters found in cells of a cube [Zhang et al., 2011], the top k representative keywords based on a local tf-idf [Ravat et al., 2008], and the proportions of topics covered by the documents of interest [Zhang et al., 2009, Mendoza et al., 2015]. In addition, systems that integrate keyword search with OLAP [Ben-Yitzhak et al., 2008, Simitsis et al., 2008] have been developed to provide users with aggregate measures for OLAP data cubes on documents returned for a query by performing aggregation on the resulting document set online.

Efficient storage strategies for OLAP over nonoverlapping sets of documents have been proposed [Zhang et al., 2011], and a fully materialized approach that deals with overlapping sets has also been proposed [Jin et al., 2010], but efficient storage strategies that can handle overlapping document sets—the focus of this thesis—have not been explored. In tagged document collections, tags are treated as dimension values. Two different forms of schema can be used for determining how tags are assigned to dimensions: multidimensional schemas and single-dimensional schemas.

2.6.1 Multidimensional Schema

A multidimensional schema (MDS) stores each tag in a separate binary dimension, where 0 signifies that the corresponding tag is not assigned and 1 signifies that it is [Jin et al., 2010]. For example, if a document d_1 has tags (*Finances, Stocks*) and d_2 is tagged with (*Stocks*) only, then d_1 is stored in cell (1, 1) and d_2 is stored in cell (0, 1) of the 2-D cuboid with those two dimensions. This cuboid can answer the query $Finances \vee Stocks$ by aggregating cells (1, 1), (0, 1), and (1, 0) together, where, for this small example, the cell (1, 0) is empty. By having a separate dimension for each tag, we can ensure that aggregations performed on a cuboid do not double count any documents. Storing a data cube for MDS is a challenge when there are many tags.

2.6.2 Single-dimensional Schema

A single-dimensional schema (SDS) stores all tags in one dimension. The dimension can take on a value ranging from 1 to T, where T is the number of unique tags in the collection. This approach works well in situations where each document is assigned only a single tag.

Zhang et al. [Zhang et al., 2011] used this approach for organizing a collection of documents into nonoverlapping cells and developed a partial materialization scheme on top of it.

To avoid having a separate dimension for each tag, as is the case with MDS, Jin et al. [Jin et al., 2010] explored using SDS for storing documents with multiple tags. Unfortunately, using SDS can result in the same document being assigned to multiple cells, which is problematic when the cells in a cuboid are aggregated. Continuing with the example above, because there is only one "tags" dimension, cell(*Finances*) stores d_1 and cell(*Stocks*) stores both d_1 and d_2. In this situation, simply adding the counts for cell(*Finances*) and cell(*Stocks*) to count the number of results for the query $Finances \vee Stocks$ will result in double counting d_1. We adopt the solution to this problem developed by Jin et al., namely storing document membership information for each cell, so that when multiple cells are aggregated, cell overlaps can be detected and compensations applied. Jin et al. use a full materialization on a small data set and focus on the union operation only; optimizing conjunctive queries involving overlapping cells has not been considered.

2.7 Topic Modeling

A topic model is defined as a generative probabilistic model that uses a few distributions over a vocabulary to describe a document collection. Techniques based on Latent Dirichlet Allocation (LDA) [Blei et al., 2003] and the Correlated Topic Model (CTM) [Blei and Lafferty, 2007] have been effective in modeling how text in documents is generated. Topic modeling techniques have also been used for recommending tag assignments to documents [Bi and Cho, 2013, Krestel et al., 2009], as explained in Section 2.1.

As described by Blei and Lafferty [Blei and Lafferty, 2005],

> LDA assumes that the words of each document arise from a mixture of topics. The topics are shared by all documents in the collection; the topic proportions are document-specific and randomly drawn from a Dirichlet distribution. LDA allows each document to exhibit multiple topics with different proportions, and it can thus capture the heterogeneity in grouped data that exhibit multiple latent patterns.

The LDA model assumes that an N-word document is created with the following generative process, which is depicted in Figure 2.1 [Blei et al., 2003]:

1. Draw topic distribution $\theta_d|\alpha$ from $Dir(\alpha)$

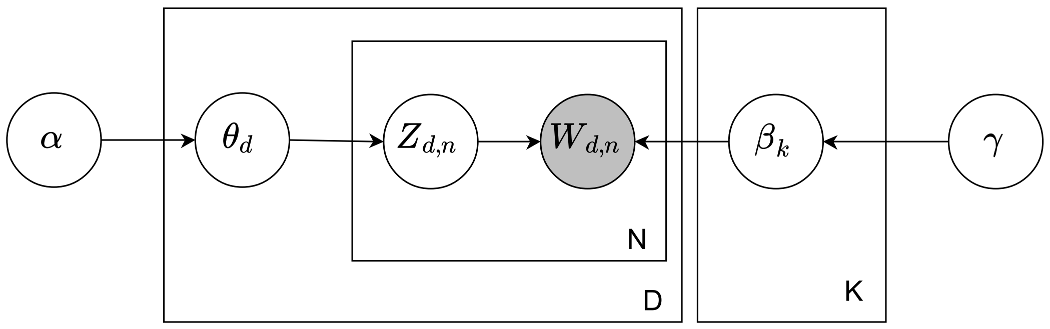


Figure 2.1: LDA generative model

2. For $n \in \{1, ..., N\}$:
   (a) Draw topic assignment $Z_{d,n}|\theta_d$ from Mult(θ_d)
   (b) Draw word $W_{d,n}|\{Z_{d,n}, \beta_{1:K}\}$ from Mult($\beta_{Z_{d,n}}$)

where K is the total number of topics, $\beta_{1:K}$ defines the word distribution for each of the K topics that is drawn from the Dirichlet distribution defined by γ, θ_d is the topic distribution of document d that is drawn from the Dirichlet distribution defined by α, $Z_{d,n}$ is the chosen topic for the n^{th} word and is drawn from the multinomial distribution defined by θ_d, and $W_{d,n}$ is the word generated.

CTM has a generative process that is very similar to LDA, except that the topic proportions assigned to each document θ_d are drawn from the logistic-normal distribution (obtained by drawing η from multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$ and then performing a logistic transformation on it) which, unlike the LDA model, captures correlations between topics assigned to documents [Aitchison and Shen, 1980]. CTM assumes that an N-word document is created with the generative process that is illustrated in Figure 2.2 and is identical to the LDA process shown above, except replacing step 1 with the following [Blei and Lafferty, 2007]:

1. Draw $\eta_d|\{\mu, \Sigma\}$ from $\mathcal{N}(\mu, \Sigma)$ and let $\theta_d = \frac{exp\{\eta_d\}}{\sum_i exp\{\eta_i\}}$

where μ is a K-dimensional mean vector and Σ is a $K \times K$ covariance matrix that together define a multivariate normal distribution $\mathcal{N}$.

Because it can model some dependencies among topics, CTM is more precise in modeling topics and generating terms than is LDA. As explained by Blei and Lafferty [Blei and Lafferty, 2005],

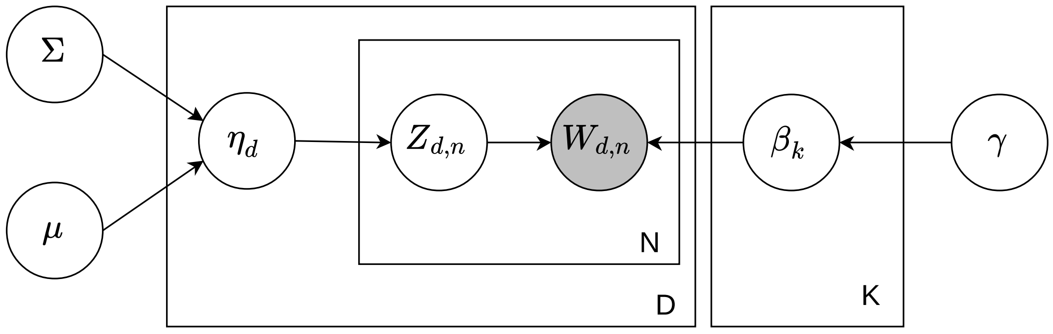


Figure 2.2: CTM generative model

> this limitation [of the LDA model] stems from the independence assumptions implicit in the Dirichlet distribution on the topic proportions. Under a Dirichlet, the components of the proportions vector are nearly independent; this leads to the strong and unrealistic modeling assumption that the presence of one topic is not correlated with the presence of another.

In addition to CTM, there exist other topic models that generate correlated topics, such as the discrete infinite logistic-normal distribution model [Paisley et al., 2012], the generalized Dirichlet distribution model [Caballero et al., 2012], pachinko allocation model [Li and McCallum, 2006], and nonparametric pachinko allocation model [Li et al., 2007]. However, CTM is simpler than the other approaches. There were no publicly available software implementations of these other models at the time of our analysis, but once they become available, they are worth consideration.

Chapter 3

Document Collections

To motivate the design of our proposed index, we evaluate document collections from two different domains: the New York Times Annotated Corpus (the NYT) [Sandhaus, 2008] and the ACM Digital Library (ACM) [White, 2001]. PubMed is used as a held out collection for testing purposes only, and we do not analyze it here.

3.1 New York Times Annotated Corpus

The NYT collection includes 1.8 million articles spanning 20 years. The collection has 1 million tags that cover many different facets, such as people, places, companies, and descriptors, and multiple tags can be assigned to each article. Out of the various types of tags contained in the collection, we consider only the tags found in the general online descriptors, which are the ones that correspond to the text found in the articles. Figure 3.1(NYT) shows a tag assignment for a single document found in the NYT collection. In our analysis we consider only the major tags, those that have been assigned to at least 200 documents, yielding 1,015 such tags that are applied to 1.5 million documents.

The tagging patterns exhibited by a document collection affect system design choices and determine whether any of the previously developed OLAP materialization strategies can be applied. We analyzed the tagging pattern for the NYT using measures adopted from analyzing tagging patterns in folksonomies such as Delicious [Cattuto et al., 2009]. These measures capture the frequency of tags appearing in the collection, the distribution of tag counts per document, and the amount of co-occurrence between the 10 most frequent tags and other tags.

Figure 3.1: An article from a) NYT with corresponding general online descriptors assigned to it, and b) ACM with corresponding category and keyword tags assigned to it

a) NYT

Article headline
Stocks drop in Tokyo
General online descriptors
Stock prices and trading volume
Stocks and bonds
Finances
Prices (Fares, Fees and Rates)

b) ACM

Title of article		
The complex dynamics of collaborative tagging		
Categories and subject descriptors		
H.5.3 [Group organisational interfaces]: collaborative computing		
I.2.4 [Artificial intelligence]: knowledge representation		
Keywords		
Tagging	Del.icio.us	Power laws
Complex systems	Emergent semantics	Collaborative filtering

The plot of tag frequencies is shown in Figure 3.2(NYT). The frequencies are normalized by the count of the most popular tag, which happens to be *Finances*, with a count of 142 thousand documents. At the other end of the spectrum, the least frequent tag (with our cutoff) has 201 documents. When the tag frequencies are sorted in descending order, the distribution resembles Zipf's law.

The number of tags assigned per document is shown in Figure 3.3(NYT). This ranges from 34% of the documents being assigned just one tag to a few documents having 43 tags, with 2.7 tags per document on average. Figure 3.4(NYT) shows the amount of document overlap between each of the 10 most popular tags and all the other tags, with the other tags shown in descending order by their frequency of co-occurrence. The presence of multiple tags per document and the high co-occurrence among the tags produce many nonempty document sets that match the conjunction of multiple tags.

3.2 ACM Digital Library

The much smaller ACM collection contains 66 thousand abstracts of articles organized with categories, general terms, and keywords. In our analysis we consider the categories and keywords tags only, since there are only 16 general terms available. Figure 3.1(ACM) shows an instance of a tag assignment for a single article found in the ACM collection. Since this collection is so much smaller than the NYT, we include all the tags with at least five occurrences in our analysis, resulting in 9,098 tags that satisfy this criterion.

The plot of tag frequencies is shown in Figure 3.2(ACM). Again, the tags are normalized by the count of the most popular tag, which happens to be *H.5.2*, with a count of 2,144 documents. The least frequent tag with our cutoff has been assigned to five documents. Just as for the NYT, the distribution of tag frequencies resembles Zipf's law.

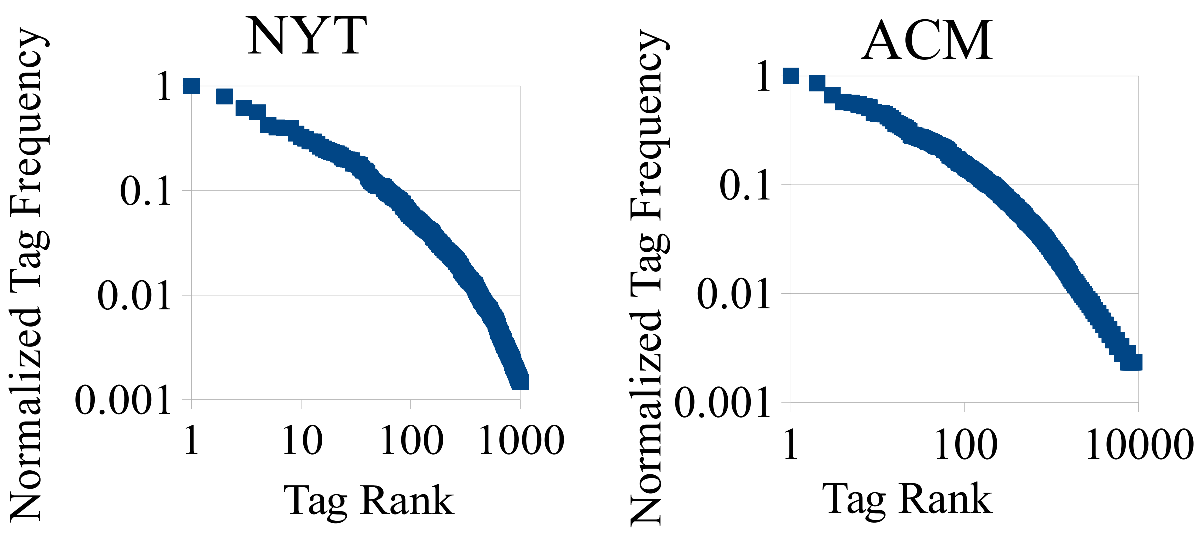


Figure 3.2: Tag assignment frequency in a) NYT and b) ACM

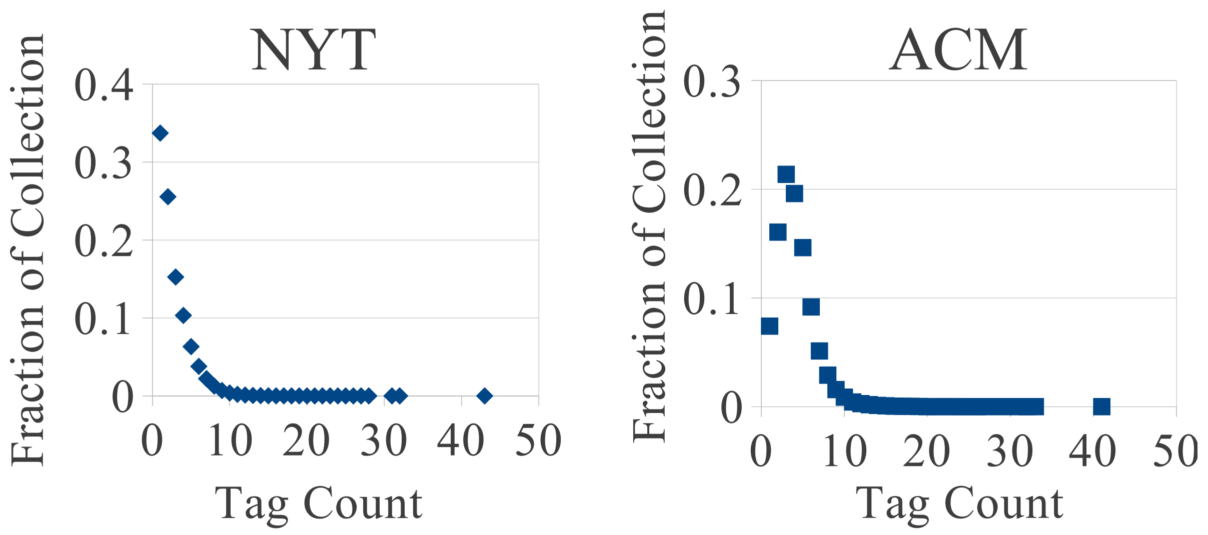
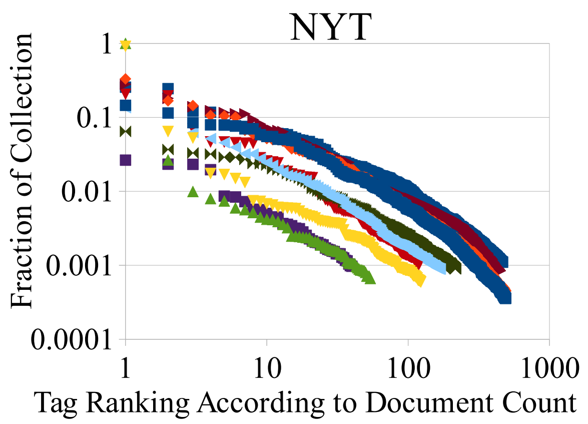


Figure 3.3: Distribution of the number of tags per document for a) NYT and b) ACM



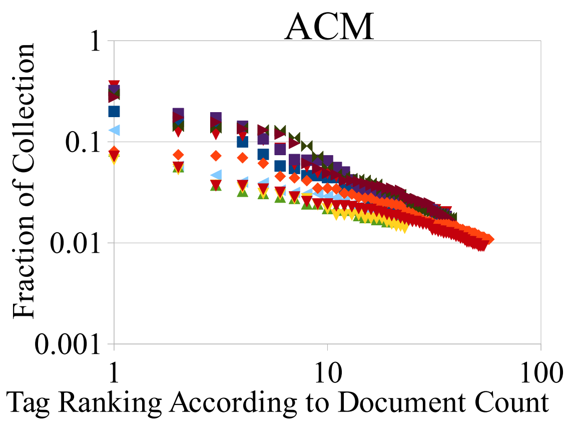


Figure 3.4: Tag co-occurrence frequencies for the 10 most frequent tags in a) NYT and b) ACM

The number of tags assigned per document ranges between 1 and 41, as shown in Figure 3.3(ACM). The mean number of tags per document is 4.1 (when both keywords and categories are combined). As was true for the NYT collection, the distribution has a very wide range, with the majority of documents having fewer than 10 tags. Since there is a higher mean number of tags per document in the ACM, we expect a larger number of tag conjunctions to produce nonempty document sets.

Figure 3.4(ACM) shows the proportion of documents that have one of the 10 most popular tags and some other tags. The shape of the ACM graph is somewhat similar to that for the NYT, but its magnitude is significantly higher, showing that the ACM tags are more inter-correlated.

3.3 Deeper Analysis of Tagging Patterns

For additional insight into the tagging patterns exhibited by the NYT and ACM collections, two more properties are analyzed. The first property refers to the order of tag co-occurrences, while the second property refers to the similarity between sets in the collections.

Table 3.1: Number of conjunctions of n tags that contribute to high multi-way co-occurrence for the NYT and ACM, with threshold limits of 50 for the NYT and 5 for the ACM.

n	1	2	3	4	5	6	7	8	9	10	11	12	Total
NYT	1,015	16,448	20,905	12,217	5,289	2,401	1,152	493	151	27	2		60,100
ACM	9,098	14,262	5,280	3,860	3,700	3,199	2,390	1,520	776	297	79	13	44,474

3.3.1 Higher Order Tag Co-occurrence

We define a collection to have a *high n-way co-occurrence* among its tags if the number of documents having n tags in common is greater than k for many different combinations of tags. Such document sets are of interest because their document count may be too high to have the set centroid calculated online, and this measure indicates the dimensionality of cuboids that need to be materialized in order to answer queries on tags efficiently.

The tag co-occurrence measures we have described for the NYT and ACM collections show that there is significant correlation among various pairs of tags, but it does not tell us if there is also high n-way co-occurrence for $n > 2$. The threshold used for determining whether an n-way co-occurrence is high should be set to be the size of a document set for which it would be efficient to calculate a summary online. For our experiments, we have chosen $k = 50$, which is quite appropriate for the NYT. However, the ACM is significantly smaller, and we wish to test how well our approaches scale up; therefore, we have chosen to use $k = 5$ for the ACM in order to expose its tagging structure in more detail.

Table 3.1 shows that there are many surprisingly large tag sets where the corresponding document count is above the threshold. If tags were assigned to each document independently of other tags, then the chance that the intersection of more than four tags would result in a document set of size greater than the threshold would be low. The multi-way correlation that exists between tags has a big impact on the number of cells that need to be materialized.

3.3.2 Overlap Between Document Sets

The similarity between two sets of documents S_a and S_b can be quantified by evaluating the size of their symmetric difference $|S_a \triangle S_b|$. We consider two sets as similar if $|S_a \triangle S_b|$ is less than or equal to the predefined threshold k, which is again set to 50 for the NYT and 5 for the ACM.

Table 3.2: Percent overlap of cells for given query lengths

NYT

	1	2	3	4	5	6	7	8	9	10	11
1		3	1	0	0	0	0	0	0	0	0
2			55	38	34	40	53	63	72	82	100
3				94	90	90	95	99	100	100	100
4					100	100	100	100	100	100	100

ACM

	1	2	3	4	5	6	7	8	9	10	11	12
1		5	9	15	16	14	11	9	6	3	1	0
2			71	87	96	98	100	100	100	100	100	100
3				98	100	100	100	100	100	100	100	100
4					100	100	100	100	100	100	100	100

Let $\mathcal{D}$ be a document collection and let $\mathbb{A} \subseteq 2^{\mathcal{D}}$ and $\mathbb{B} \subseteq 2^{\mathcal{D}}$ be sets of document sets. We define the *k-overlap* $O_k(\mathbb{A}, \mathbb{B})$ between $\mathbb{A}$ and $\mathbb{B}$ as follows:

$$O_k(\mathbb{A}, \mathbb{B}) = \frac{|\{S_a \in \mathbb{A} \mid \exists S_b \in \mathbb{B} \wedge |S_a \triangle S_b| < k\}|}{|\mathbb{A}|} \tag{3.1}$$

By applying $O_k(\mathbb{A}, \mathbb{B})$ to pairs of sets reflected in Table 3.1, we can compute the percentage of overlap between document sets corresponding to the conjunction of i tags and document sets corresponding to the conjunction of j tags. This is summarized in Table 3.2, where the i^{th} column shows the percentage of overlap with sets of size j ($j < i$) and all entries are 100 for $i > j \geq 4$. Thus, cells corresponding to grouping by five tags or more have a very high similarity score to cells corresponding to four or fewer tags. Examination of the tags reveals that often after the first few tags are assigned to a document, additional tags are semantically similar to ones already present. Like Blei and Lafferty [Blei and Lafferty, 2005], we hypothesize that the tags assigned to documents reflect the inclusion of certain topics and that a document covers a limited number of correlated topics. As more tags are assigned to a document, eventually there will be multiple tags referring to the same topic.

The high similarity between document sets in a collection can be exploited to reduce the number of centroids that are materialized.

3.4 Significance of Tagging Patterns

The analyses of tagging patterns found in the NYT and ACM reveal important challenges associated with such document collections. First, since there are many popular tags, some of which are assigned to as many as 140,000 documents, it is infeasible to compute set centroids for corresponding document sets online. Instead, it is necessary to precompute the data in order to guarantee satisfactory response times. Second, since many documents are assigned multiple tags and there is high correlation among the tags, it is reasonable to expect users to issue queries using tag conjunction. The more tag combinations that can be meaningfully conjoined, the more possible sets of documents exist about which a user can enquire. Third, surprisingly many tags can appear in a conjunctive query that yields a large document set. This leads to the existence of many sets for which centroids need to be computed; far more than one would expect if tags were assigned randomly. Materializing centroids for all these combinations is infeasible.

Fortunately, there is a lot of overlap between sets of documents of different query lengths. Because of this overlap, we can achieve considerable savings in storage cost by developing a suitable partial materialization strategy that can scale to large document collections with large numbers of tags. We explore this further in Section 5.3.

3.5 Vocabulary Properties of Collections

As described in Section 2.4.6, there are two approaches to storing the vocabulary (features) used by the documents in a collection: *local* and *global*. With *local* vocabulary a different set of features are used for each category, and the features used to describe a document depend on the categories it belongs to (determined by the tags assigned to it). With *global* vocabulary all categories share a common set of features that are used to represent all documents.

The number of unique terms and their distribution in a document collection affects the amount of space required for storing centroids of document sets and which vocabulary storage technique will be most appropriate to use for representing the data. Before analyzing document content in the NYT and ACM collections, we perform standard preprocessing steps to clean and reduce the size of the vocabulary. As part of the preprocessing we remove all terms that contain numbers, convert all terms to lower case, remove stop words (33 terms), apply a Porter Stemmer, and remove all terms with global term frequency below five.

Table 3.3: Effect of the number of top k features taken from each tag on the size of global vocabulary

k	**Global vocabulary size**	
Features/Tag	**NYT**	**ACM**
All	238,065	13,015
1,000	127,589	13,001
800	104,290	13,000
500	64,358	13,000
358	46,576	13,000
100	17,128	12,999
50	10,666	12,994
25	6,766	12,960
5	2,479	10,318
2	1,410	5,930
1	867	3,635

After the above five preprocessing steps are applied, 238,065 unique terms remain in the NYT collection, and 13,015 in the ACM collection. Storing a full vocabulary for document collections consumes a lot of space and at the same time for any one tag only a small fraction of the total vocabulary is relevant. The uninformative terms can be removed from the vocabulary by applying a feature selection technique such as *mutual information*, and only keeping the most important terms for each tag. Table 3.3 shows the effect on the size of global vocabulary of picking the top k features of the sub-collection corresponding to each tag. For example, keeping the top 500 terms for each tag based on mutual information results in storing 64,358 terms for the NYT collection, whereas keeping the top term for each tag in the ACM collection results in storing a global vocabulary of 3,635.[1]

Since the terms used in the global vocabulary are chosen from tag specific terms, the majority of them show up as features for only a small number of tags, as the plots in Figure 3.5 indicate. The plots correspond to the cumulative distribution function of the proportion of all terms that are found within tag vocabularies in the NYT and ACM collections. The plot of the NYT uses a vocabulary size of 64,358, and the plot of the ACM uses a vocabulary size of 3,635. In the NYT collection roughly 80% of terms are

[1]Often, the same term is picked as a representative for multiple tags, which leads to a smaller than expected global vocabulary. For example, in the NYT collection the term ‘tax’ is chosen as a representative feature for seven different tags that include “Taxation”, “Federal Taxes (US)”, “Income Tax”, “Tax Evasion”, “Tax Credits”, “Sales Tax”, and “Excise Taxes”.

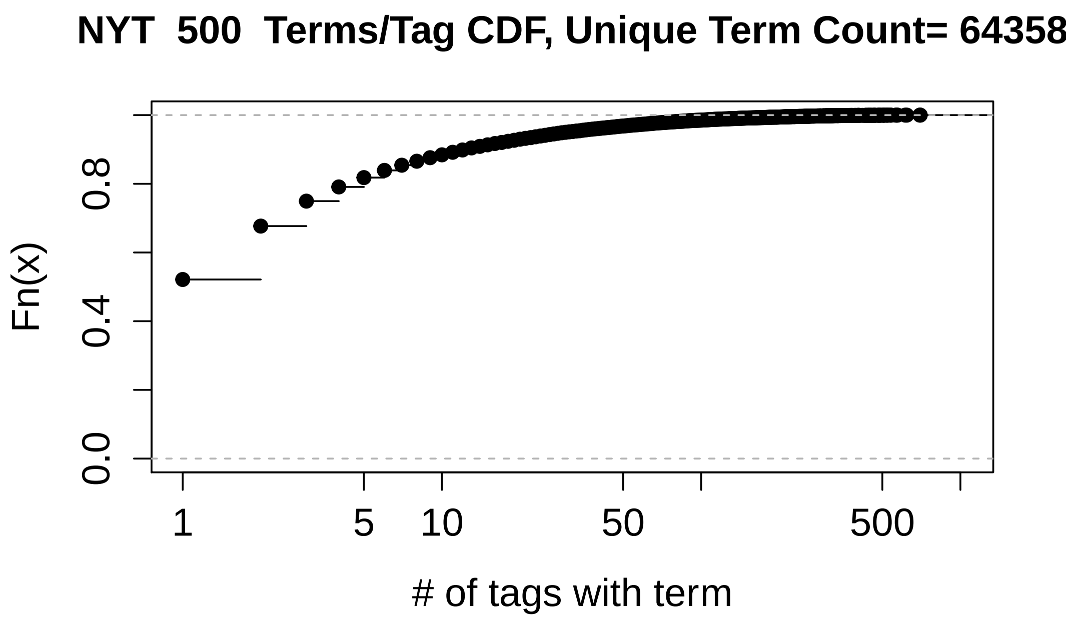
found in five or fewer tags out of 1,015 available, while in the ACM collection around 80% of terms are found in fewer than three out of 9,098 available tags. This suggests, that in both collections, the vocabulary is localized, such that, for different tags there are different terms that are relevant.

The expressiveness of a shared global vocabulary is examined in Figures 3.6 for the NYT and 3.7 for the ACM, which show the average number of top k tag specific terms that are available when a global vocabulary storage technique is used. The global vocabulary for the NYT consists of 2,479 terms (formed by taking the top 5 tag-specific features from each of the tags found in the collection) and for the ACM consists of 3,635 terms (formed by taking the top tag-specific feature from each of the tags found in the collection). The global vocabulary of the NYT on average contains only 49.3 of the top 100 tag specific terms (coverage of 49.3%) and 161.7 of the top 500 tag specific terms (coverage of 32.3%), while the global vocabulary of the ACM on average contains 23.7 of the top 50 tag specific terms (coverage of 47.4%) and 181.1 of the top 500 tag specific terms (coverage of 36.2%). From these results, we can conclude, that the global vocabulary representation of the NYT and ACM collections provides poor coverage of the tag specific features.

3.6 Summary

In this chapter, we examined the tagging patterns found in the NYT and ACM collections, and we highlighted the challenges of developing an infrastructure that can support efficient access to summaries over such collections. In Chapter 4, we develop a generative model that attempts to reproduce and generalize the tagging patterns we have observed in the NYT and ACM. The observations serve to inform the development of materialization strategies that support efficient storage and access to summary information for sets of documents found in the NYT and ACM collections, as well as other collections that contain similar tagging patterns.

In this chapter, we also examined how the term vocabulary (of text found in the body of each article) is distributed throughout different tag-defined sets in the NYT and ACM collections, and we described the challenges of using a shared global vocabulary to support the storage of rich summaries. In Chapter 6, we develop alternative vocabulary storage approaches to address these observed challenges.



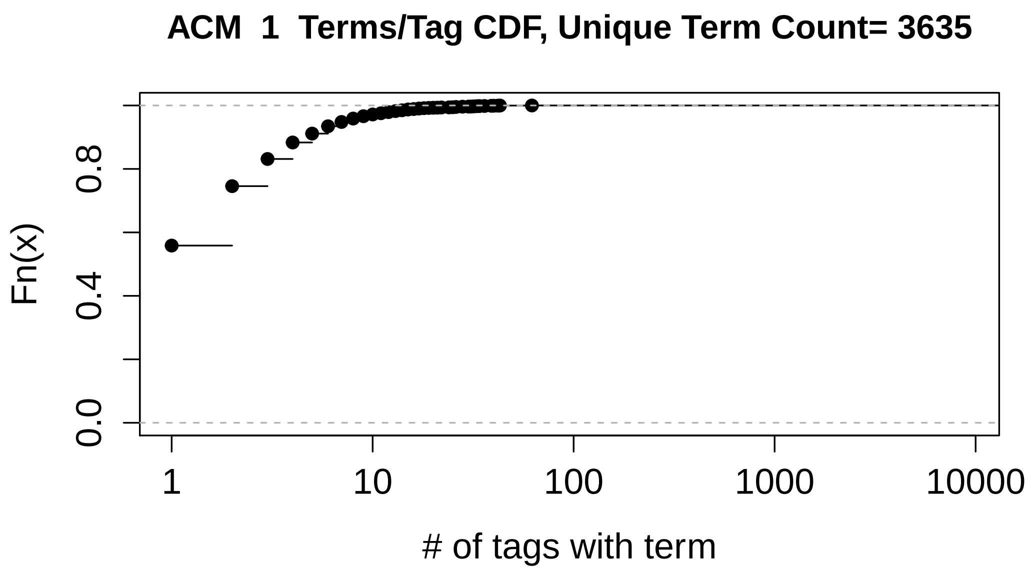
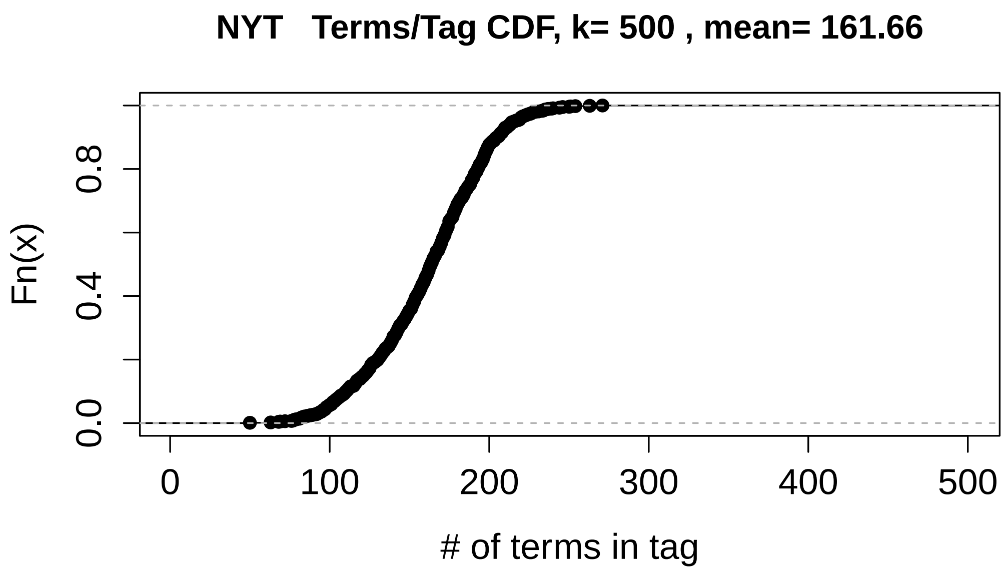


Figure 3.5: Percentage of terms that occur in fewer than X tags a) NYT and b) ACM



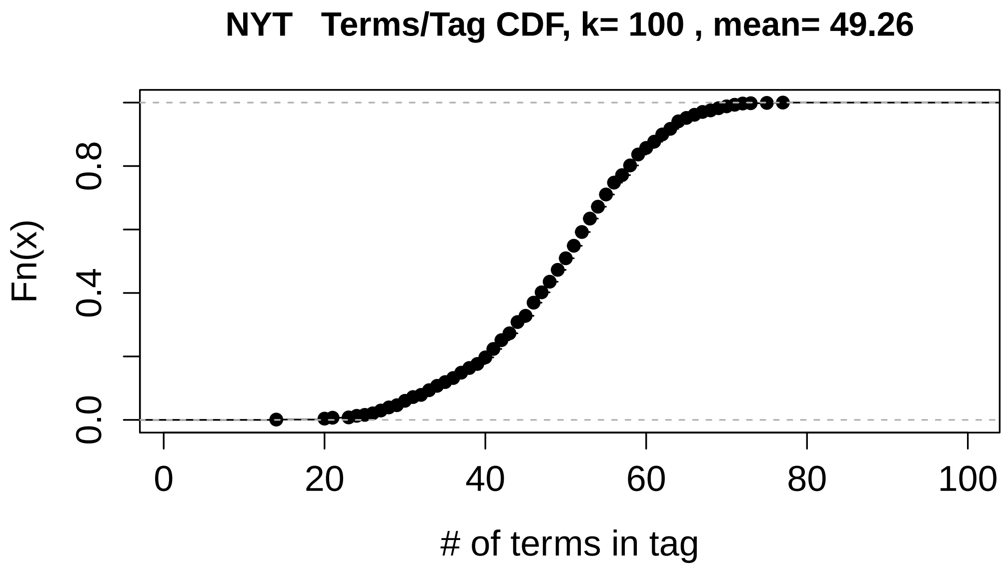
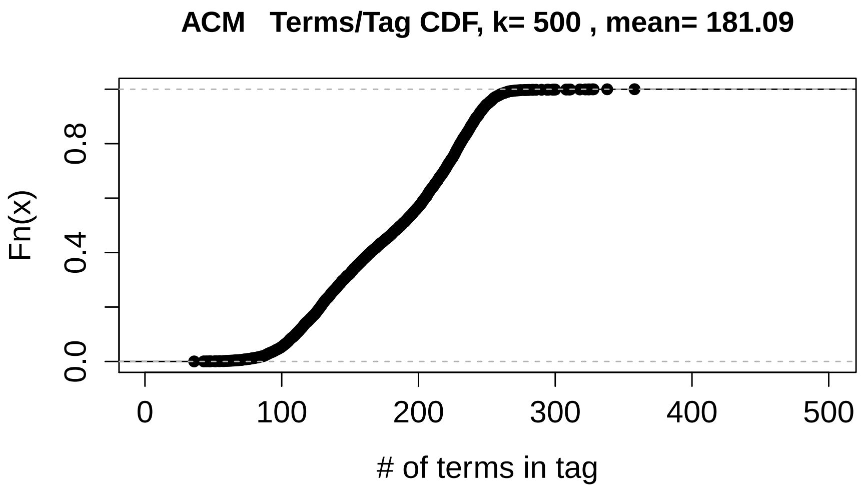


Figure 3.6: Percentage of tags which contain less than X Number of terms when using a global vocabulary of size 2,479 (where top 5 terms come from each tag) in the NYT



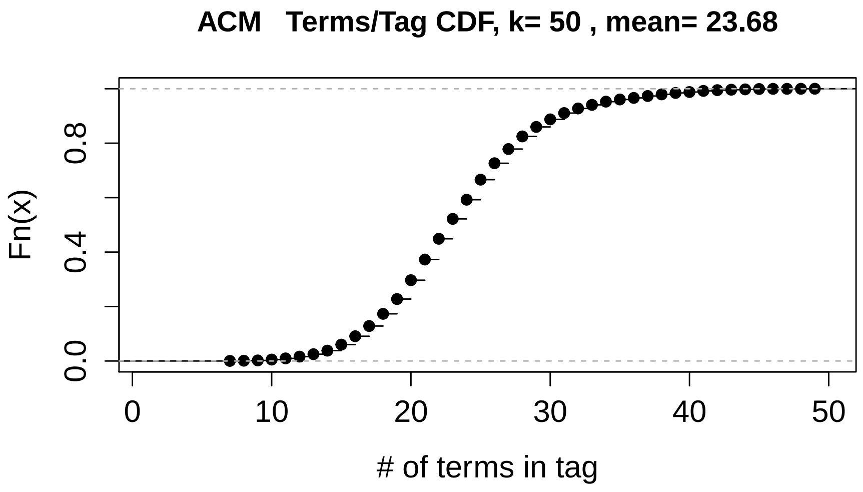


Figure 3.7: Percentage of tags which contain less than X Number of terms when using a global vocabulary of size 3,635 (where 1 term come from each tag) in ACM

Chapter 4

Generative Tagging Model

The motivation for deriving a generative tagging model for the tagging patterns observed in the NYT and ACM collections is so that we can synthesize document collections with a wide variety of realistic tagging patterns, on which we can evaluate our materialization strategies.

With a generative tagging model we can establish a controlled environment for evaluating the effectiveness of our materialization strategies on multi-tagged document collections with various properties. This can provide us with a good understanding of the types of collections that our infrastructure can support. In addition, by having a generative tagging model, against which we have tested our proposed materialization strategies, we can characterize a class of multi-tagged document collections that can benefit from our work: Any collection that can be fitted to our proposed generative tagging model and whose parameters are within our tested range, is a good candidate to benefit from our proposed materialization strategies.

Thus, building on earlier work [Bi and Cho, 2013, Blei and Lafferty, 2005, Blei and Lafferty, 2007], we describe a generative tagging model based on correlated topic modeling. We adapt the technique to generate tags instead of document content and then fit the model to our two prototypical document collections. Afterwords, the variables of the fitted model are further generalized by fitting them to appropriate distributions. This allows us to approximate the resulting tag generation model in place of using the inferred latent variables from the NYT and ACM directly. The result is a model that allows us to generate a wide variety of realistic tagged document collections.

4.1 Algorithm for Tag Generation

Topic models have the simplified view that the text in documents results from terms being randomly sampled from the topics that are implicitly present in the document. The tag patterns observed in documents can be viewed as an extension to this model: Instead of generating terms to include in a document's content, we wish to generate the document's tags. To accomplish this, we need to adapt the text generation algorithm to accommodate the properties of tags:

1. Tags should be drawn only from topics that have a significant presence (i.e., exceeding a given threshold F).

2. Topic models for text assume that each document consists of N words, where N is large. The number of tags assigned per document is much smaller, varies from document to document, and depends on the set of contributing topics for each document. A document in the NYT has on average 2.7 tags while a document in the ACM has on average 4.1 tags.

3. Words in the text of a document can occur multiple times, but documents do not have repeated instances of the same tag assigned to them.

With respect to point 1, the topic distributions generated for individual documents (θ_d) when using correlated topic modeling tend to include many topics that are assigned a non-zero proportion of the probability density function [Caballero et al., 2012]. This is not problematic when generating words that are to be included in a document's text content. However, since tags are assigned to a document only when topics have a significant presence in the document's text, we constrain tag generation to derive only from topics that have a distributional proportion exceeding the threshold value F.

If we did not apply a threshold, the presence of a large number of topics with small proportions would result in the generation of undesirable tags. For example, if we assume that a generative model for the ACM collection has 70 topics, then any document generated with this model has a tag distribution θ_d that allocates a nonzero value to each of the 70 topics. Typically, when a document is generated, a couple of topics are assigned a large value while the remaining topics are assigned small nonzero values. However, the CTM generative model has a tendency of producing 'small nonzero' values that are not small enough and can cause problems in aggregate. For instance, if the generated θ_d has a distribution where topic one is assigned a value of 0.4, topic two a value of 0.26, and the remaining 68 topics a value of 0.005 each. Then with this distribution, there is a very high likelihood (34 %) that the generative model will assign to a document an undesirable tag from one of the 68 topics that are non-representative of the document. By only drawing from topics that exceed a threshold, we prevent the undesirable tags from being drawn. We found experimentally that setting the threshold F to 9% performs well in selecting the 'right amount' of contributing topics.

Elaborating on point 2, the number of tags in documents in both the ACM and NYT collections is fairly small, but varies widely. We hypothesize that the number of tags assigned to a document depends on both the number of contributing topics found in the text and which specific topics contribute to the tagging. We model this observation by drawing a tag count for each of the significant topics present in the document and then aggregating all the individual counts as is formally defined in Algorithm 1. Since, in our model, the number of tags assigned to a document depends on the number of contributing topics, it was necessary to have the threshold F set appropriately (which was determined by tweaking the F parameter until the generative model produced a total tag count for the synthetic collection that matched the total number of tags found in the original collection).

Algorithm 1 Get tag count for document

```
**Input:** *Filtered* topic distribution $\theta_d$; Tag count distribution $\tau_t$ for each potential topic
**Returns:** Tag count
$N = 0$
**for all** $t \in \theta_d$ with non-zero proportion assigned **do**
    Draw $c$ from Mult($\tau_t$)
    $N = N + c$
**end for**
**return** $N$
```

For the third point above, it is straightforward to avoid repeating tags by merely sampling for terms without replacement. Note that in our approach, we do not generate a text first and then choose words from that text to serve as tags. Rather we generate tags directly from the model. As a result, because the terms available in the vocabulary of tags are discriminatory, there is no need to rely on inverse document frequency to eliminate common terms.

Algorithm 2 formally describes tag generation. For the correlated topic model (CTM), the topic simplex S, from which a topic distribution is chosen, is derived from a logistic–normal distribution (Fig. 2.2). (If the latent Dirichlet allocation model were to be used instead, the simplex S would have a Dirichlet distribution.)

Algorithm 2 Tag generation algorithm

```
**Input:** Topic simplex $S$ over $K$ topics; Topic threshold $F$; Collection size $D$; Tag distribution for topics $\beta = \{\beta_1, ..., \beta_K\}$; Tag count distribution for topics $\tau = \{\tau_1, ..., \tau_K\}$
**Returns:** Set of assigned tags for each document
Document set $\mathcal{D} \leftarrow \emptyset$
for $d \in 1..D$ do
    Draw topic distribution $\theta_d$ from $S$
    for all $t \in \theta_d$ do
        $\theta[t] \leftarrow (\theta_d[t] > F)\ ?\ \theta_d[t]\ :\ 0$
    end for
    $\theta \leftarrow \theta / \sum \theta[t]$
    Tag count $N_d \leftarrow getTagCountForDocument(\theta, \tau)$
    Tag set $T_d \leftarrow \emptyset$
    for $n \in 1..N_d$ do
        Draw topic $Z$ from Mult$(\theta)$
        Draw tag $W$ from Mult$(\beta_Z)$ *without replacement*
        $T_d \leftarrow T_d \cup \{W\}$
    end for
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{T_d\}$
end for
return $\mathcal{D}$
```

4.2 Fitting the Generative Model

The generative model is instantiated by fixing the following parameters: the probability distribution of the K topics $P(Topic)$ captured by μ and the pairwise topic covariance matrix Σ, the probability distribution of tags in each of the K topics $P(Tag|Topic)$ captured by β_k, and the tag count per document distribution $P(Tag\_Count|Document)$ captured by the topic filter threshold F and the tag count per topic distribution $P(Tag\_Count|Topic)$ represented by τ_k.

We fit the CTM[1] generative model against the NYT and ACM collections and then evaluate how well the learned model can mimic the tagging behavior found in the two collections. Due to the cost of training topic models with CTM and its inability to handle very large collections [Caballero et al., 2012], only a subset of the ACM and NYT collections are used for training and validation purposes. For the NYT collection we have taken a uniform sample of 10% of the original collection for analysis and a model with 30 topics learned based on half of that data (5% of original collection) and validated on the other half. For the ACM we used 20% of the collection to generate a model with 70 topics, again learning and validating on disjoint halves of that sample. For both collections, the number of topics for the generative model was chosen by examining generative models with a wide range of topic counts and selecting the model that generated tagging patterns with high order of co-occurrence among the tags and had a high likelihood of generating tagging patterns observed in held out documents. For the ACM collection we examined generative tagging models with topic counts ranging between 45 and 150, while for the NYT collection we examined generative tagging models with topic counts ranging between 20 and 30. Since Paisley et al. [Paisley et al., 2012] also considered 30 topics in their analysis of the NYT, it suggests that our choice of the topic count to use for our NYT generative tagging model is reasonable. Our goal is to have a generative tagging model that creates complex tagging patterns with high order of co-occurrence, and not to identify the most accurate and intuitive topics for the collection, and so, picking an optimal number of topics is not critical to our work. The other parameters used for fitting the two models are specified in Table 4.1.

4.3 View of the Inferred Topics

The distributions of topic means for the generative models learned for the NYT and ACM collections are shown in Figures 4.1 and 4.2 respectively, and the corresponding topic

[1] We use the CTM implementation provided by Blei at http://www.cs.princeton.edu/~blei/ctm-c/.

Table 4.1: Setting parameters for CTM

Parameter	**CTM Value**
em max iter	500
var max iter	500
cg max iter	500
em convergence	10^{-4}
var convergence	10^{-6}
cg convergence	10^{-5}
covariance estimate	mle

correlation matrices are shown in Figure 4.3 for the NYT and Figure 4.4 for the ACM. In the correlation matrices, the green colour corresponds to positive correlations, the red colour to negative correlations, and the size of the boxes to the strength of correlation. In Figure 4.5 we show the correlation matrix for 30 topics in the NYT collection inferred from text as was observed by Paisley et al. [Paisley et al., 2012]. (The two NYT correlation matrices were derived from different subsets of the NYT collection.) The correlations between the 30 topics inferred from tags in the NYT are significantly weaker than the correlations between the top 30 topics inferred from text. Since each document has many fewer tags than words in its text, it is reasonable to see weaker correlations for topics inferred from tags. We do not see this discrepancy as a problem with the modeling because the topics that were identified with a higher correlation had a semantic relationship between their tags, and the tags that belonged to topics with higher correlation co-occurred together more frequently than tags that came from less correlated topics. So, although the effects of correlation were much weaker in the tag generative model, they were still present. Furthermore, by manipulating the correlation matrix, we were able to control the frequency of co-occurrence among tags that come from correlated topics in a predictable way, which makes us believe that the model is working appropriately. For example, in Table 4.9 we compare the effect of using different correlation matrices on the tagging patterns generated by the generative tagging models. We compared a generative model that uses an identity matrix as the correlation matrix (*S2*) against a generative model that uses the correlation matrix from the learned ACM model (Figure 4.4) that was boosted by multiplying all correlations by 3 (*S5*). As is demonstrated by *S5*, when there exist strong correlations among topics, more tags will co-occur with each other.

In Table 4.2 we show two pairs of correlated topics that were inferred by the CTM for the NYT collection, and in Table 4.3 we shows two pairs of correlated topics that were inferred for the ACM. For each tag we show the probability of the tag being generated by

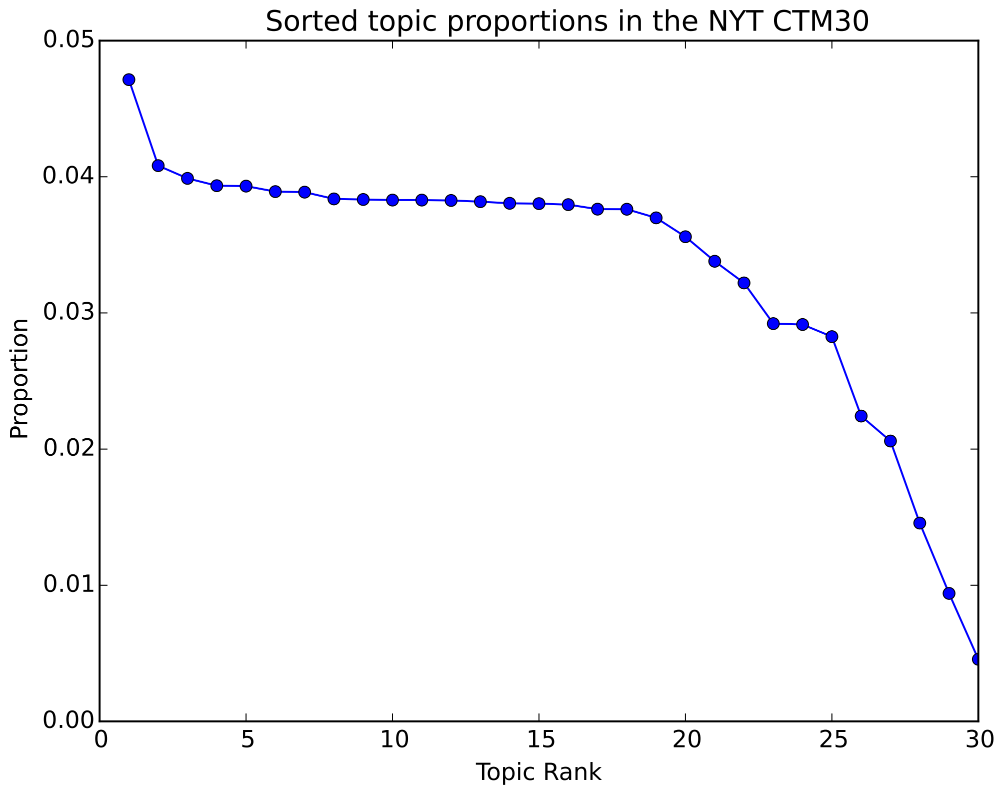


Figure 4.1: Topic distribution in the CTM model of the NYT with 30 topics

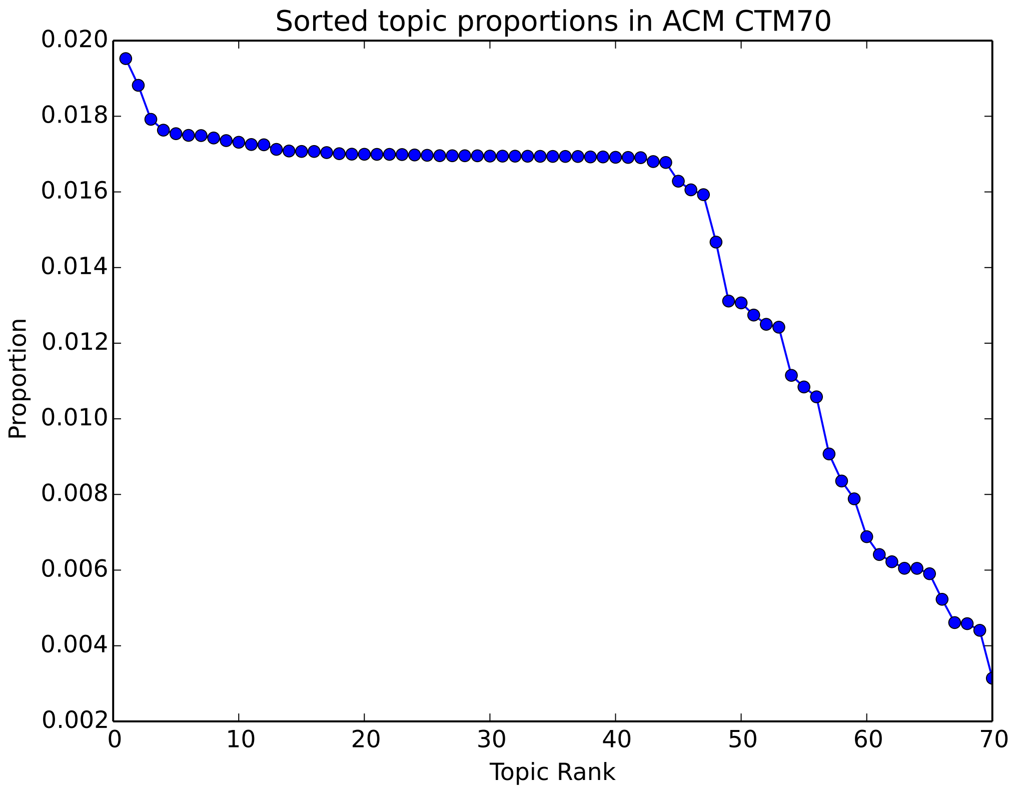


Figure 4.2: Topic distribution in the CTM model of the ACM with 70 topics

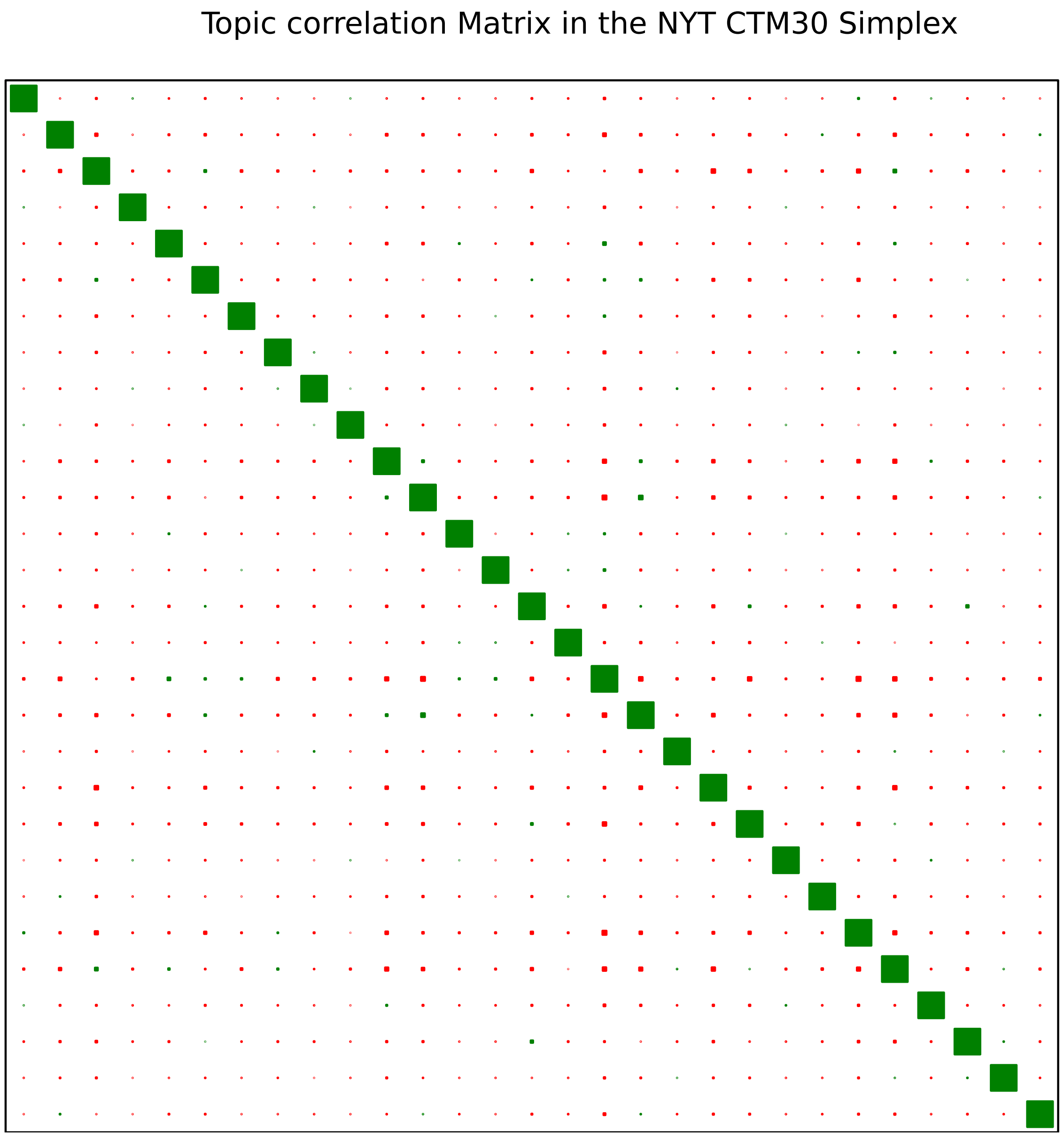


Figure 4.3: Topic correlation matrix for 30 topics in the NYT inferred from tags

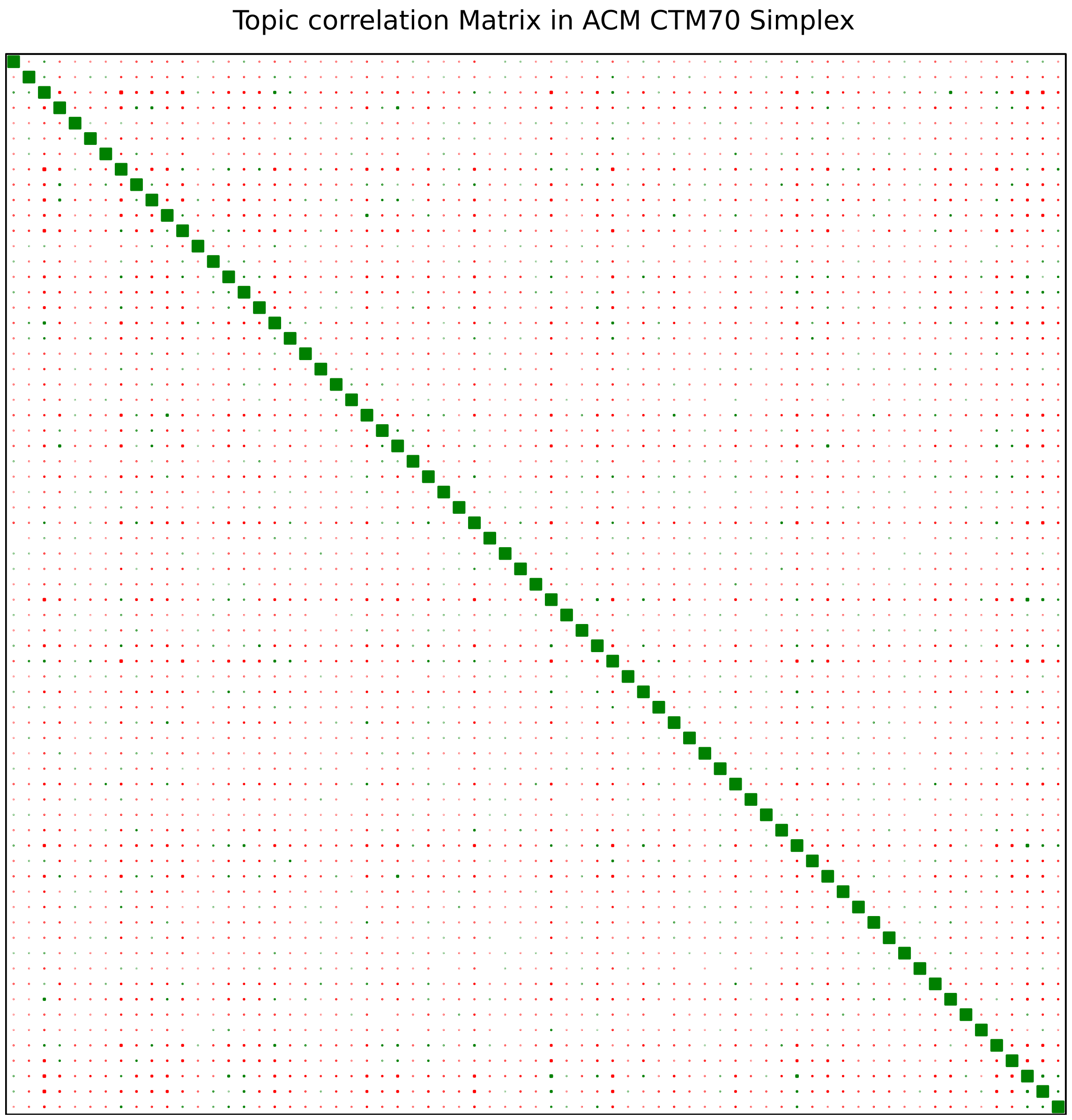
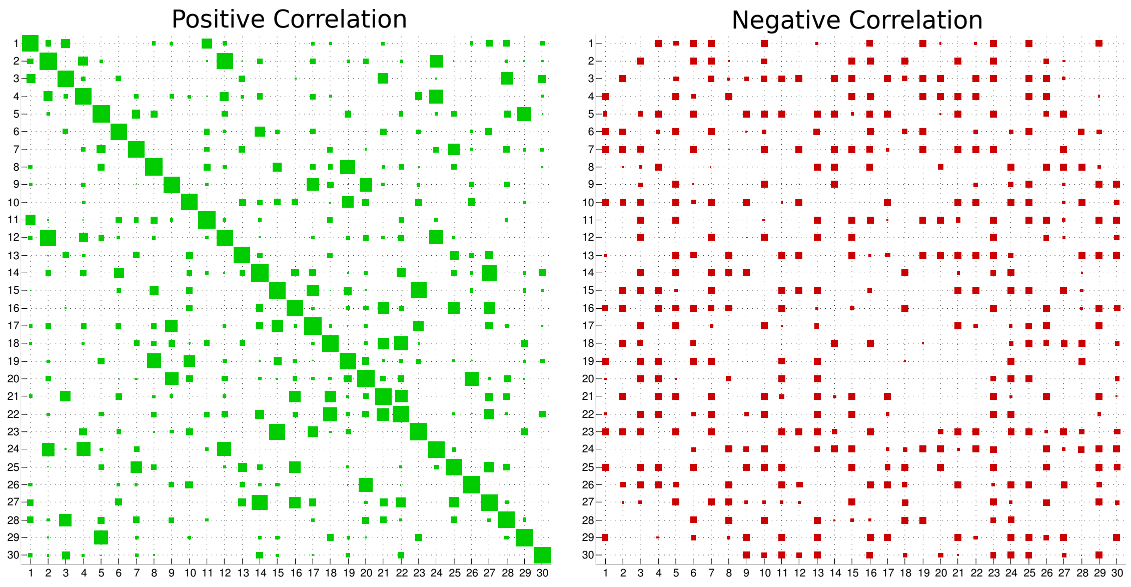


Figure 4.4: Topic correlation matrix for 70 topics in the ACM inferred from tags



Topic 1:	campaign, democratic, candidate, republican, election, voter, political, presidential, vote, party
Topic 2:	game, victory, second, score, third, win, team, play, season, lose
Topic 3:	president, executive, chief, vice, name, director, advertising, chairman, senior, company
Topic 4:	team, player, season, coach, game, play, football, league, contract, sign
Topic 5:	add, heat, pound, cup, oil, minute, water, large, dry, serve
Topic 6:	building, build, house, space, site, project, construction, area, foot, plan
Topic 7:	drug, patient, treatment, study, disease, risk, health, treat, cancer, cause
Topic 8:	economy, economic, percent, growth, increase, government, states, economist, price, rate
Topic 9:	police, officer, arrest, man, charge, yesterday, official, crime, drug, release
Topic 10:	share, company, stock, buy, percent, investment, acquire, sell, investor, firm
Topic 11:	budget, tax, cut, increase, taxis, state, plan, propose, reduce, pay
Topic 12:	shot, point, play, game, hit, ball, night, shoot, player, put
Topic 13:	computer, internet, information, site, technology, system, software, online, user, program
Topic 14:	art, artist, museum, exhibition, painting, collection, gallery, design, display, sculpture
Topic 15:	government, political, country, international, leader, soviet, minister, states, foreign, state
Topic 16:	book, story, write, novel, author, life, woman, writer, storey, character
Topic 17:	attack, kill, soldier, bomb, bombing, area, official, report, group, southern
Topic 18:	song, sing, band, pop, rock, audience, singer, voice, record, album
Topic 19:	market, stock, price, fall, trading, dollar, investor, trade, rise, index
Topic 20:	trial, lawyer, charge, prosecutor, case, jury, guilty, prison, sentence, judge
Topic 21:	play, movie, film, star, actor, character, theater, role, cast, production
Topic 22:	dance, stage, perform, dancer, company, production, present, costume, theater, performance
Topic 23:	peace, israeli, palestinian, talk, palestinians, territory, arab, leader, visit, settlement
Topic 24:	guy, thing, lot, play, feel, kind, game, really, little, catch
Topic 25:	science, theory, scientific, research, human, suggest, evidence, fact, point, question
Topic 26:	court, law, state, legal, judge, rule, case, decision, appeal, lawyer
Topic 27:	image, photograph, picture, view, photographer, subject, figure, paint, portrait, scene
Topic 28:	report, official, member, commission, committee, staff, agency, panel, investigate, release
Topic 29:	wine, restaurant, food, menu, price, dish, serve, meal, chicken, dining
Topic 30:	graduate, marry, father, degree, receive, ceremony, wedding, daughter, son, president

Figure 4.5: Topic correlation matrix for 30 topics in the NYT inferred from text along with the ten most probable words for each topic

the topic. The results appear to be intuitively acceptable.

4.4 Evaluating the Topic Model

We hypothesize that a good set of topics has been inferred when there is coherence among the top-K tags in each topic (tags are associated with a single overarching semantic concept) and the tag distributions between topics are dissimilar (i.e., there is little redundancy between topics).

4.4.1 Measuring Topic Coherence

In the ACM collection the category tags contain a coding scheme that is composed of a letter followed by one or two numbers (e.g., H.1.3), which together identify the tags' location in a hierarchical tree. Recall that we ignored this hierarchical structure when we created our learned topic model. Nevertheless, we hypothesize that tags located close to each other in the category tree tend to be more similar to each other than those that are far, and if many similar tags appear together in the same topic, the topic will be coherent. We define the similarity of two hierarchical tags $\sigma(t_1, t_2)$ as the length of the common prefix for those tags. For example, H.1.2 has similarity score 3 with itself, similarity score 2 with H.1.3, and similarity score 0 with A.1.2. Topic coherence can then be calculated using Equation 4.1.

$$\mathcal{C}(C) = \frac{1}{|C|} \sum_{T_i \in C} \frac{1}{\binom{|T_i|}{2}} \sum_{\substack{t_1, t_2 \in T_i \\ t_1 < t_2}} P(t_1|T_i) P(t_2|T_i) \sigma(t_1, t_2) \tag{4.1}$$

where C is a collection of topics, each topic T_i is composed of a distribution of category tags t, and $P(t|T)$ is the probability of category tag t appearing in topic T. Higher scores imply that the topics in the collection have more related category tags, which is expected of good topic assignments. Since keyword tags found in the ACM do not have any hierarchical data, they do not contribute towards the topic coherence score. Thus, the topic coherence score is calculated from a modified set of topics which had all the keyword tags removed from their distributions and were renormalized.

Table 4.4 shows that topics inferred for the ACM with correlated topic modeling have higher category similarity scores than topics that are generated through a random process of assigning each of the tags found in documents to a random topic. This suggests that

Table 4.2: Sample of correlated topics and their top-k tags for the NYT

Topic 11		**Topic 17**	
Tag	%	*Tag*	%
Murders and Attempted Murders	18.5	Crime and Criminals	23.6
Blacks	14.7	Demonstrations and Riots	8.8
Police	11.7	Sex Crimes	6.5
Drug Abuse and Traffic	9.1	Freedom and Human Rights	5.2
Transit Systems	5.7	Christians and Christianity	4.5
Robberies and Thefts	5.2	Prisons and Prisoners	4.1
Assaults	3.6	Sentences (Criminal)	4.0
Police Brutality and Misconduct	3.5	Child Abuse and Neglect	3.6
Subways	3.3	Religion and Churches	2.8
Violence	3.2	Fines (Penalties)	2.5
Sum	78.6	*Sum*	65.6

Topic 4		**Topic 16**	
Tag	%	*Tag*	%
Prices (Fares, Fees and Rates)	35.0	Finances	42.8
International Trade and World Market	15.0	Mergers, Acquisitions and Divestitures	10.7
Sales	13.3	Stocks and Bonds	10.6
Currency	8.5	Credit	3.6
Ratings and Rating Systems	6.4	Frauds and Swindling	3.4
Records and Achievements	5.9	Banks and Banking	2.9
Auctions	5.8	Futures and Options Trading	2.2
Small Business	1.7	Government Bonds	2.1
Discount Selling	1.5	Bankruptcies	1.9
Coffee	1.1	Executives and Management	1.7
Sum	94.2	*Sum*	81.9

Table 4.3: Sample of correlated topics and their top-k tags for the ACM

Topic 10		**Topic 23**	
Tag	%	*Tag*	%
machine learning	5.7	I.2.6:	12.3
genetic programming	5.3	G.1.6:Global optimization	5.5
J.6:Computer-aided design (CAD)	5.0	I.2.8:	4.6
genetic algorithm	3.7	evolutionary algorithms	4.0
I.2.8:Heuristic methods	3.2	F.2.1:	3.9
I.2.2:Program synthesis	3.2	I.2.8:Heuristic methods	3.6
I.2.6:Concept learning	2.6	evolutionary computation	3.5
I.2.8:Plan execution, formation, and generation	2.2	multi-objective optimization	3.1
heuristics	1.8	G.3:Probabilistic algorithms (including Monte Carlo)	3.1
I.2.6:Induction	1.7	genetic algorithms	2.8
Sum	34.5	*Sum*	46.5

Topic 18		**Topic 52**	
Tag	%	*Tag*	%
C.2.1:Wireless communication	50.8	wireless sensor networks	18.8
C.2.1:Network communications	9.7	C.2.2:Routing protocols	15.3
C.2.3:	6.6	C.2.1:Network topology	13.3
wireless ad hoc networks	3.9	adaptation	11.0
OFDM	3.1	performance evaluation	7.1
cognitive radio	2.2	C.2.2:Protocol verification	3.3
rate control	1.7	topology control	2.8
medium access control	1.3	congestion control	2.8
capacity	1.3	broadcast	2.4
ad hoc networking	0.9	data collection	1.9
Sum	81.5	*Sum*	78.8

Table 4.4: Category similarity score comparison for topics generated using the CTM, and randomly generated topics on the ACM

	Category quality score ($\mathcal{C}$)		
	Topic count	CTM	Random
ACM	70	0.1380	0.0525

the ACM collection does have some coherent latent topics into which the CTM is grouping the tags. (A similar analysis could not be performed on the NYT collection since it does not have hierarchical tags.)

4.4.2 Measuring Topic Separation

The distance between topic distributions can be evaluated using the Jensen-Shannon measure [Lee, 1999] described in Equation 4.2. We consider both the average Jensen-Shannon distance between all pairs of topics (Equation 4.3) and average minimum Jensen-Shannon distance (Equation 4.4). Table 4.5 summarizes topic distances of topics produced by the CTM generative model and topics that are generated through a random process of assigning each of the tags found in documents to a random topic. In both collections the topics inferred through the CTM model have greater separation than randomly generated topics, although the difference for the ACM collection are not as profound as for the NYT. We hypothesize that since in the ACM collection there are significantly more tags than in the NYT (9,098 in ACM vs. 1,015 in NYT), it is more likely that the randomly generated topics will have a higher average distance and average minimum distance from each other.

$$JS(q,r) = \frac{1}{2}\Big[D(q\|\tfrac{q+r}{2}) + D(r\|\tfrac{q+r}{2})\Big] \tag{4.2}$$

where

$$D(q\|r) = \sum_y q(y)\log\frac{q(y)}{r(y)}$$

$$JS_{avg}(T) = \frac{\sum_{q,r\in T, r\neq q} JS(q,r)}{\binom{\|T\|}{2}} \tag{4.3}$$

$$JS_{min}(T) = \frac{\sum_{q\in T} min_{r\in T, r\neq q}(JS(q,r))}{|T|} \tag{4.4}$$

Table 4.5: Distance between topic distributions generated by the CTM and randomly

		CTM		**Random**	
	Topic count	$JS_{avg}(T)$	$JS_{min}(T)$	$JS_{avg}(T)$	$JS_{min}(T)$
NYT	30	0.6501	0.5785	0.0394	0.0372
ACM	70	0.6675	0.5969	0.5421	0.5181

Table 4.6: Likelihood analysis on the generation of test data

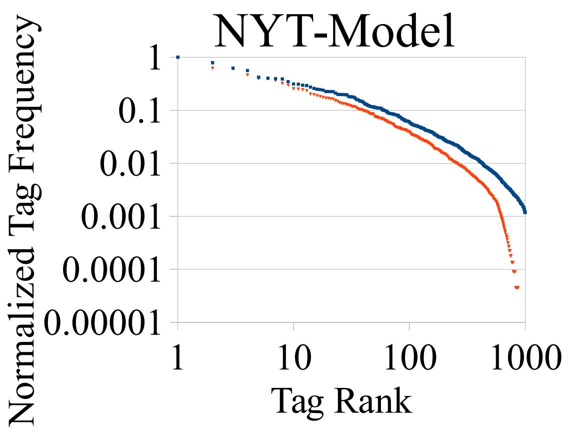
	Test type	Log-likelihood
NYT	set of held out documents	-14.88
NYT	50% tags replaced by random ones	-19.58
NYT	100% tags replaced by random ones	-21.96
ACM	set of held out documents	-28.39
ACM	50% tags replaced by random ones	-33.18
ACM	100% tags replaced by random ones	-35.80

4.4.3 Evaluating Models' Ability to Generate Correct Tagging Patterns

The learned models are evaluated by comparing the likelihood of generating all the tagging patterns observed in a set of held out documents against the likelihood of generating the tagging patterns found in sets that had the tags assigned to documents randomly. Two variants of document sets with random tag assignments are considered. In the first approach, the original set of held out documents is used, and for each document 50% of its tags are randomly selected and replaced by a new set of randomly selected tags. In the second approach, for each document in the original set of held out documents, all the tags are replaced by a new set of tags that are randomly selected with uniform probability. We conclude that the models were successfully trained to generate the tagging patterns that appear in the collections examined: Table 4.6 shows that the model is more likely to generate the tagging pattern observed in the held out documents than the tagging patterns that result from adding noise.

4.5 Properties of Tags Generated by the Models

The learned models were used to generate tag assignments, which were then compared to the original collections, with respect to the properties described in Chapter 3:



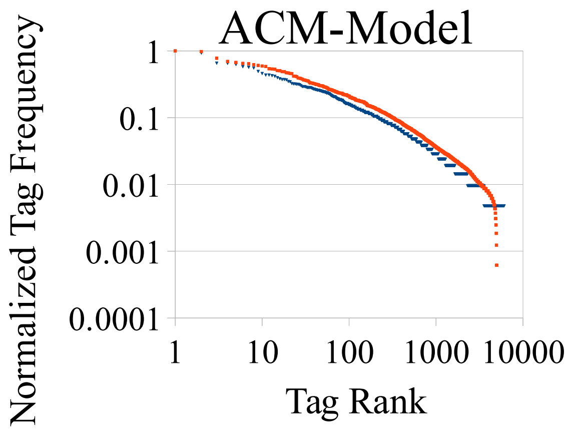
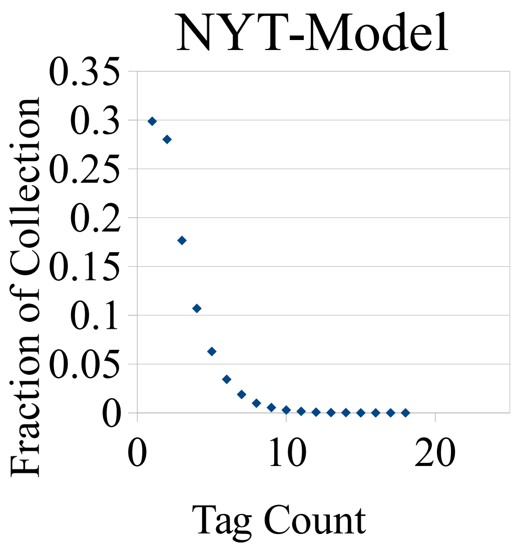


Figure 4.6: Tag assignments in generated vs. original collections a) NYT and b) ACM

1. the tag frequency distribution,
2. the tag count per document distribution,
3. the pairwise tag co-occurrence patterns,
4. the distribution of higher-order tag co-occurrence, and
5. the number of distinct tag sets with large amount of similarity.

Due to the large size of the NYT collection, tags for a smaller collection—10% of size of the NYT—were generated, and, similarly to the ACM, the threshold limit for online document aggregation is assumed to be 5 instead of 50 (the value for the whole NYT collection).

Figure 4.6 overlays the normalized tag frequency distributions observed in the original collections (Figure 3.2) with that of the learned model, showing that it tracks the original distribution fairly closely for all but the least frequent tags. The learned model has a bias towards selecting the popular tags for assignment to documents, which in turn leads to some of the less popular tags not being assigned to any documents. For the NYT collection, out of 1,015 available tags, the generative model only assigned the 744 most popular tags to the generated documents. While for the ACM collection, out of 6,109 available tags, the generative model only assigned the 4,890 tags. (The ACM generative model is capable of only generating 6,109 tags because only that many tags were observed in the training set of the generative model.)



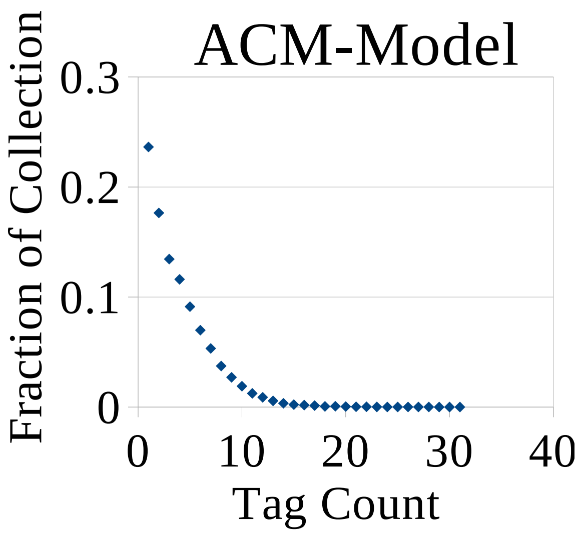


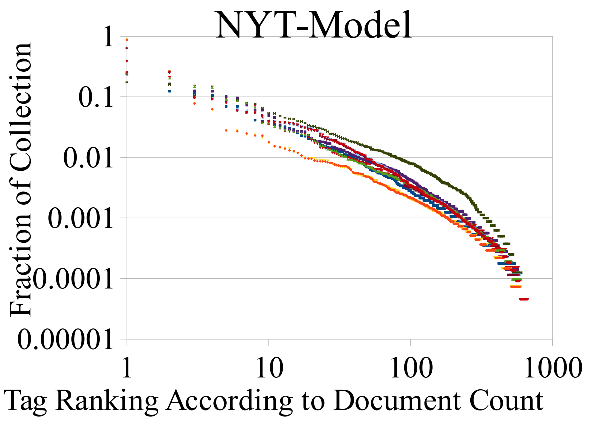
Figure 4.7: Distribution of the number of tags for synthetic tag assignment generated using CTM for a) NYT and b) ACM

The tag count per document distributions observed in the original sampled collections (Figure 3.3) can be compared to the distributions in the learned model (Figure 4.7). The learned model of the NYT has the mean tag counts per document close to the original sampled collection and the general shapes of the curves are similar. For the learned model of the ACM, the mean tag counts per document are close to the original sampled collection but the shape of the curve differs. However, a similar distribution shape can be achieved by reducing the threshold F, at the expense of increasing the mean tag count per document. All the observed distributions of tag count per document resemble the zero-truncated Poisson distribution.

Comparing the pairwise tag co-occurrence observed in the original sampled collections (Figure 3.4) to those for the learned model (Figure 4.8), shows that the top tags co-occur with many other tags and apparently resemble a Zipf-like distribution.

The amount of higher order tag co-occurrence is shown in Table 4.7 (which repeats the ACM data from Table 3.1 for convenience). The learned model includes many documents that have many tags in common, although the sets of shared common tags do not grow as large as in the original collections, especially for the ACM-Model.

Similarity among document sets in the NYT and ACM (Table 3.2) can be compared to the corresponding generated collection (Table 4.8). Although, there is some similarity



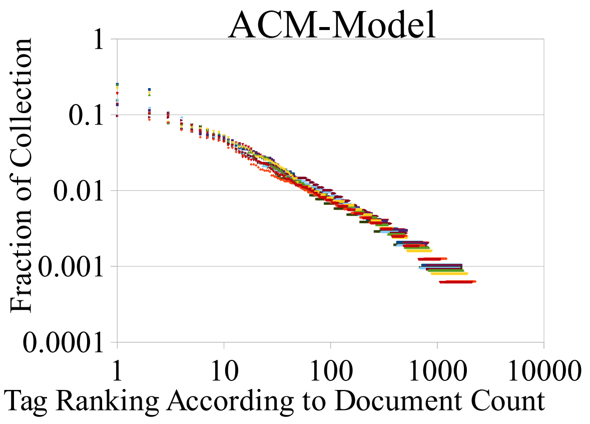


Figure 4.8: Tag to tag co-occurrence graph of the 10 most frequent tags for synthetic collection generated by CTM in a) NYT and b) ACM

Table 4.7: Number of conjunctions of n tags that contribute to high multi-way co-occurrence (with threshold limit 5)

		1	2	3	4	5	6	7	8	9	10	11	12	Total
NYT	Fragment	1,015	18,657	26,199	16,995	7,733	3,423	1,757	951	423	130	24	2	77,309
	Model	744	15,795	18,524	7,244	1,646	310	62	11	1				44,337
ACM	Original	9,098	14,262	5,280	3,860	3,700	3,199	2,390	1,520	776	297	79	13	44,474
	Model	4,890	15,675	2,836	213	7								23,621

Table 4.8: Percentage overlap of cells corresponding to two different query lengths using $O_k(\mathbb{A}, \mathbb{B})$ in the a) NYT-Model and b) ACM-Model

NYT-Model

	1	2	3	4	5	6	7	8	9
1		0	0	2	8	27	58	82	100
2			20	12	27	56	81	91	100
3				70	54	73	87	91	100
4					96	94	95	100	100

ACM-Model

	1	2	3	4	5
1		0	0	0	0
2			16	2	0
3				76	71
4					100

observed between different tag intersection levels in the synthetic collections, it is not as profound as in the original collections, especially for the ACM-Model.

In the development of the generative tagging models, our goal was not to create perfect generative models for the NYT and ACM collections, but rather to create a generative tagging model that produces synthetic collections that exhibit realistic tag assignment patterns. As a result, the various differences between the generated tag assignment patterns and the tag assignments found in original collections, do not indicate a failure of the approach; the tag assignment patterns generated do have many properties that are similar to those observed in the original data.

4.6 Additional Synthetic Models

The tag distribution $\beta_{1:K}$ for each of the K topics in both the NYT and ACM collection is drawn from $Dir(\gamma)$ and so topics for a synthetic collection can be generated by drawing from the Dirichlet distribution with an appropriate γ. Higher γ values will result in tag distributions where many tags have similarly low probability of occurrence, which will result in little co-occurrence among tags. A low γ value will generate tag distributions that have few dominant tags with high probability, which will produce a large amount of tag co-occurrence among the popular tags.

An instance of a document collection is analogous to an instance of a document: just as each document is assumed to be generated by drawing a topic distribution from a simplex (with a logistic-normal distribution for the CTM model), a distribution of topic means μ for a whole document collection can be drawn from a simplex with Dirichlet distribution $Dir(\varphi)$ (since we do not care about correlation when drawing topic distributions for collections).

The topic covariance matrix Σ is used to control which topics will be drawn together when the topic distribution for a document is drawn. When Σ is a diagonal matrix then the generative model becomes equivalent to Latent Dirichlet Allocation.

In the inferred models for both the NYT and ACM, the tag count distributions are best approximated by zero-truncated Poisson distributions, as specified by Equation 4.5 where λ varies between 0.01 and 2.27 for the NYT-Model and between 0.06 and 2.86 for the ACM-Model. The Kolmogorov-Smirnov test was used on top of the observed tag count distributions for topics found in both the NYT and ACM to evaluate if they were drawn from zero-truncated Poisson distribution. For the 30 topics found in the NYT, 21 of them could not be rejected by the Komogorov-Smirnov test; while for the 70 topics in the ACM, 42 of them could not be rejected. In addition, in the original LDA paper by Blei et al.[Blei et al., 2003], the word count for each document is drawn from Poisson distribution. This suggests that the approach of choosing the number of tags assigned to a document by drawing it from a zero-truncated Poisson distribution is reasonable.

$$P_{ztp}(k,\lambda) = \frac{\lambda^k}{(e^\lambda - 1)k!} \tag{4.5}$$

The variability of the λ parameters defining the tag count per topic distributions in both collections suggests that the number of tags assigned to a document depends on the topics it covers.

The distribution of tag counts per topic can be modeled by first taking the set of tags assigned to each document and determining the most likely source topic for each tag using Equation 4.7. The identified topics for each document are then aggregated to produce topic counts per document.

$$P(topic_x|tag_y, doc_z) = \frac{P(tag_y|topic_x)P(topic_x|doc_z)}{P(tag_y|doc_z)} \tag{4.6}$$

$$\underset{topic_x}{\arg\max}\ P(tag_y|topic_x)P(topic_x|doc_z) \tag{4.7}$$

By drawing the topic distribution vector μ from $Dir(\varphi)$, tag distribution β for each topic from $Dir(\gamma)$, tag count distribution for each topic from $P_{ztp}(k, \lambda)$, setting a topic threshold F, and inducing various types of topic correlations, we can generate realistic synthetic collections with various characteristics. For example, we can reduce the topic filter threshold to induce more topics per document and, in turn, generate more tags per document. We can modify the entries in the topic covariance matrix to increase/decrease the number of times certain topics co-occur in documents or increase the mean of the tag counts in the topic distribution to induce more tags per topic and thus more tags per document. Finally, we can lower the γ parameter that defines the Dirichlet distribution from which tag distribution for topics is drawn to increase the dominance of the most popular tags.

By testing our partial materialization techniques (described in Chapter 5) on synthetic collections generated by the proposed tag generative model, we can observe how robust the partial materialization techniques are to collections with different characteristics. With that purpose in mind, we use the parameters specified in Table 4.9 to generate synthetic collections. At the bottom of the table, we display the resulting effects on tag distributions.

4.7 Conclusions

We developed a generative tagging model that generates tagging patterns that mimic those observed in the NYT and ACM collections. The synthetic datasets produced by this generative model are used for evaluating the performance of various materialization strategies examined in Chapter 5.

For deriving the generative tagging model we used the correlated topic model, which was able to reproduce the tagging patterns observed in the NYT and ACM collection reasonably well. We did not focus on deriving a perfect fit of the generative models to the observed collections but instead aimed at a model that can create a synthetic collection with the observed properties. Challenges that have been pointed out in the literature were also observed by us: The biggest one is that many parameters, such as the number of topics, need to be set for the model before it is fitted. The discrete infinite logistic–normal distribution model [Paisley et al., 2012] does not require the user to specify the number of topics, which may make it easier to learn the generative model. The CTM has a tendency to assign a nonzero proportion to many topics, which is undesired and not a problem with the generalized Dirichlet distribution [Caballero et al., 2012]. As a result, it may be worthwhile in the future to investigate how other generative models perform in reproducing the tagging patterns found in the document collections.

Table 4.9: Input parameters and resulting tagging patterns for synthetic collections

	NYT Model	ACM Model	S1	S2	S3	S4	S5	S6
Topic count (K)	30	70	30	70				
Unique tag count	1,015	6,109	1,015	6,109				
Document count	145,701	66,041	145,701	66,041				
Topics' tag distrib.	Inferred from NYT	Inferred from ACM	*Dirichlet* ($\gamma = 0.005$)	*Dirichlet* ($\gamma = 0.001$)				*Dirichlet* ($\gamma = 0.0005$)
Topic means distrib.	Inferred from NYT	Inferred from ACM	*Dirichlet* ($\varphi = 20$)					
Topic covariance matrix,	Inferred from NYT	Inferred from ACM	No correlation				Triple tag correl.	No correlation
Tag count per topic	Inferred from NYT	Inferred from ACM	*zero trunc. Poisson*, $\lambda \in$ [0.01,2.27]	*zero trunc. Poisson*, $\lambda \in$ [0.06,2.86]		*zero trunc. Poisson*, $\lambda \in$ [1.06,3.86]	*zero trunc. Poisson*, $\lambda \in$ [0.06,2.86]	
Min. presence req'd for tag	9%	9%	9%	9%	7%	9%		

Materialization threshold (k=5)								
Query length	NYT Model	ACM Model	S1	S2	S3	S4	S5	S6
1	744	4,890	617	1,532	1,583	1,758	1,669	1,040
2	15,795	15,675	28,020	14,259	21,776	54,204	28,692	16,258
3	18,524	2,836	36,296	6,266	10,759	103,073	49,951	9,606
4	7,244	213	7,196	1,455	2,962	95,189	38,030	2,849
5	1,646	7	540	133	348	51,913	14,127	605
6	310		15	6	12	15,784	2,201	76
7	62					2,895	95	7
8	11					449		
9	1					70		
10						5		
Total	44,337	23,621	72,684	23,651	37,440	325,340	134,765	30,441

Chapter 5

Partial Materialization

In this chapter, we address the problem of supporting efficient calculation of centroids for sets of documents defined through Boolean queries over tags in multi-tagged document collections. We work under the assumption that full materialization is not feasible due to limited storage space and that users require query results within a fixed amount of time. To address this problem, we require a partial materialization strategy that can support high dimensional data with hundred of thousands of dimensions and perform well on our expected query workload. Since none of the existing approaches satisfy our requirements, we introduce three novel fine-grained partial materialization strategies. We demonstrate that our approaches enable efficient storage and retrieval of centroids in multi-tagged document collections such as the NYT and ACM. The performance of the proposed partial materialization strategies are evaluated against competing approaches on both real and synthetic collections to demonstrate their superiority. In Chapter 7, we illustrate how these strategies for deriving centroids can be incorporated into a faceted browsing system that addresses the larger problem introduced in Chapter 1.

5.1 Storing Document Member Sets

To support intersection and union operations on cells with overlapping sets of documents, the IDs of member documents need to be accessible for any given cell. Rather than storing document membership information for cells, even for those that are materialized, we store the list of defining tags with each materialized cell and rely on the postings list of documents for each tag to find the corresponding set of document IDs (Section 1.1.4). By storing each tag's postings list as a compressed bitmap with Word-Aligned Hybrid (WAH) encoding [Lemire et al., 2010], we require 17.8MB for the NYT collection, which has 1.5 million articles, and 1.7MB for the ACM collection, which has 66 thousand articles. With such a low memory footprint, it is feasible to have the *tag* postings lists stored in memory. Thus, this approach conserves storage space and efficiently supports finding the set of documents associated with a cell.

5.2 Granularity of Materialization

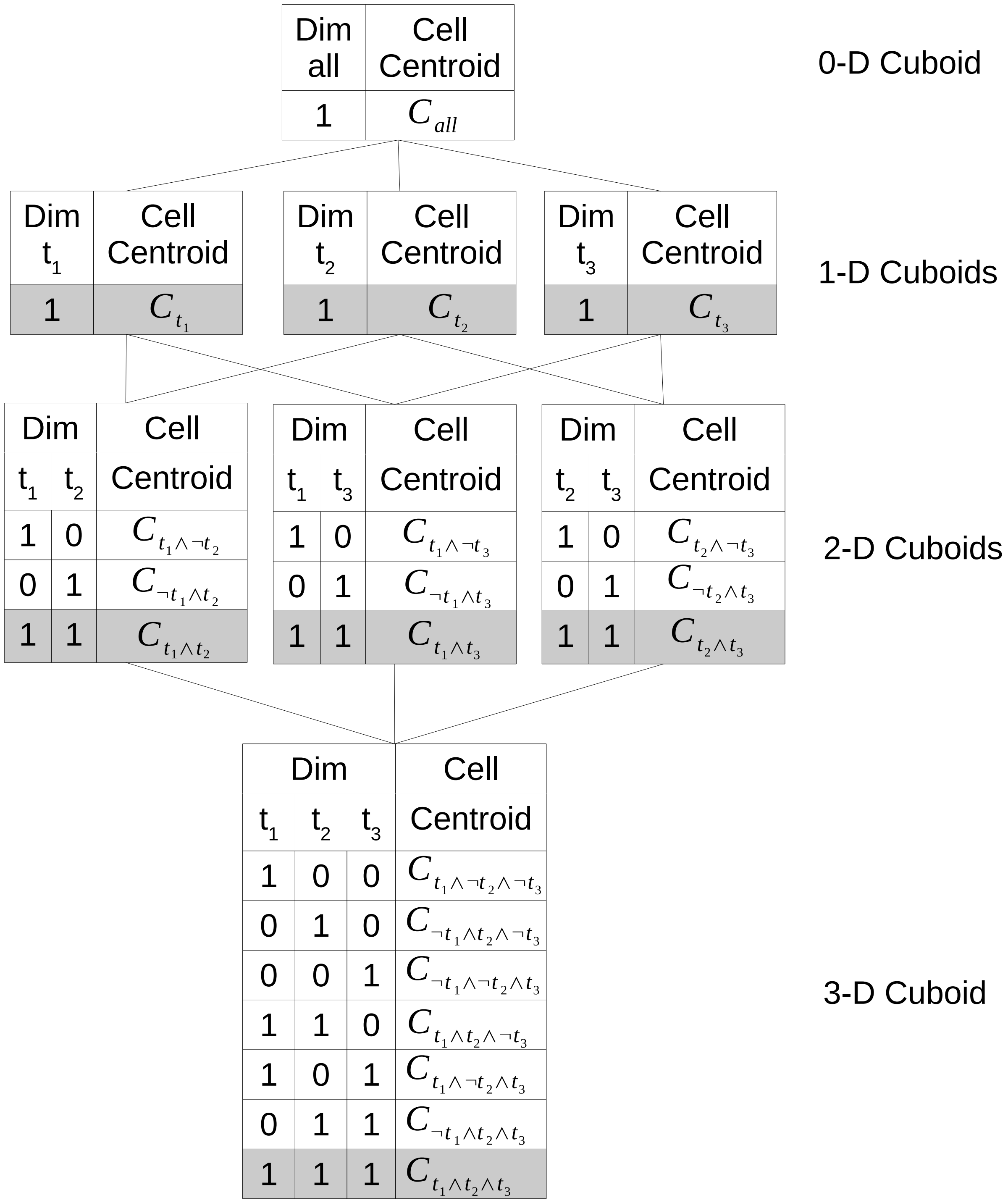
As part of the materialization process we need to select the data that will be stored. As explained in Section 2.5, the selection decisions have traditionally been made at the granularity of whole cuboids. However, one could also make the selection decision at a finer level of granularity that involve individual cells. We compare the two materialization approaches and show that by using individual cell materialization instead of full cuboid materialization, we can save on both the storage cost and the cost of servicing the expected query workload.

Before describing the two approaches in detail, we first illustrate with the help of Figure 5.1 which set of data found in a lattice of cuboids for a set of three tags $(t_1, t_2.t_3)$ will be materialized by each approach. Depending on which queries need to be supported, the full cuboid materialization strategy will materialize an appropriate subset of cuboids found in the lattice. For example, all three 2-D cuboids might be materialized to support all queries that involve two tags, or the 3-D cuboid may be materialized to support all queries that involve three tags. On the other hand, the individual cell materialization strategy only materializes the cells that are marked grey.

5.2.1 Full Cuboid Materialization

A d-D cuboid, which can answer all queries that involve the subset of dimensions found in it, has $\prod_{i=1}^{d} n_i$ cells, where n_i is the number of unique values in dimension i. Since in a multidimensional schema each tag dimension has only two values (0 and 1), a d-D cuboid has 2^d cells. However, the cell that has all dimensions set to zero (i.e., none of the tags present) is not required when evaluating queries with at least one positive term (Section 1.1.2), and so only $2^d - 1$ cells need be stored.

An example of a 3-D cuboid is shown in the first two columns of Table 5.1a. The cuboid consists of seven cells, one for each assignment of three tags (except for the cell that has all dimensions set to zero). Although a d-D cuboid can answer all queries involving any



Dim all | Cell Centroid
1 | C_{all}
0-D Cuboid
Dim t_1 | Cell Centroid
1 | C_{t_1}
Dim t_2 | Cell Centroid
1 | C_{t_2}
Dim t_3 | Cell Centroid
1 | C_{t_3}
1-D Cuboids
Dim t_1 t_2 | Cell Centroid
1 0 | $C_{t_1 \wedge \neg t_2}$
0 1 | $C_{\neg t_1 \wedge t_2}$
1 1 | $C_{t_1 \wedge t_2}$
Dim t_1 t_3 | Cell Centroid
1 0 | $C_{t_1 \wedge \neg t_3}$
0 1 | $C_{\neg t_1 \wedge t_3}$
1 1 | $C_{t_1 \wedge t_3}$
Dim t_2 t_3 | Cell Centroid
1 0 | $C_{t_2 \wedge \neg t_3}$
0 1 | $C_{\neg t_2 \wedge t_3}$
1 1 | $C_{t_2 \wedge t_3}$
2-D Cuboids
Dim t_1 t_2 t_3 | Cell Centroid
1 0 0 | $C_{t_1 \wedge \neg t_2 \wedge \neg t_3}$
0 1 0 | $C_{\neg t_1 \wedge t_2 \wedge \neg t_3}$
0 0 1 | $C_{\neg t_1 \wedge \neg t_2 \wedge t_3}$
1 1 0 | $C_{t_1 \wedge t_2 \wedge \neg t_3}$
1 0 1 | $C_{t_1 \wedge \neg t_2 \wedge t_3}$
0 1 1 | $C_{\neg t_1 \wedge t_2 \wedge t_3}$
1 1 1 | $C_{t_1 \wedge t_2 \wedge t_3}$
3-D Cuboid

Figure 5.1: Lattice of cuboids.

Table 5.1: **a)** 3-D cuboid for tag set $\{t_1, t_2, t_3\}$ **b)** $I(T)$ for tag set $\{t_1, t_2, t_3\}$

Dim			**Cell**	**Computation**
t_1	t_2	t_3	**centroid**	**from I(T) cells**
1	0	0	$C_{t_1 \wedge \neg t_2 \wedge \neg t_3}$	$C_{t_1} - C_{t_1 \wedge t_2} - C_{t_1 \wedge t_3} + C_{t_1 \wedge t_2 \wedge t_3}$
0	1	0	$C_{t_2 \wedge \neg t_1 \wedge \neg t_3}$	$C_{t_2} - C_{t_1 \wedge t_2} - C_{t_2 \wedge t_3} + C_{t_1 \wedge t_2 \wedge t_3}$
0	0	1	$C_{t_3 \wedge \neg t_1 \wedge \neg t_2}$	$C_{t_3} - C_{t_2 \wedge t_3} - C_{t_1 \wedge t_3} + C_{t_1 \wedge t_2 \wedge t_3}$
1	1	0	$C_{t_1 \wedge t_2 \wedge \neg t_3}$	$C_{t_1 \wedge t_2} - C_{t_1 \wedge t_2 \wedge t_3}$
1	0	1	$C_{t_1 \wedge t_3 \wedge \neg t_2}$	$C_{t_1 \wedge t_3} - C_{t_1 \wedge t_2 \wedge t_3}$
0	1	1	$C_{t_2 \wedge t_3 \wedge \neg t_1}$	$C_{t_2 \wedge t_3} - C_{t_1 \wedge t_2 \wedge t_3}$
1	1	1	$C_{t_1 \wedge t_2 \wedge t_3}$	$C_{t_1 \wedge t_2 \wedge t_3}$

Dim			**Cell**	
t_1	t_2	t_3	**centroid**	**Source**
1	*	*	C_{t_1}	1-D cuboid
*	1	*	C_{t_2}	1-D cuboid
*	*	1	C_{t_3}	1-D cuboid
1	1	*	$C_{t_1 \wedge t_2}$	2-D cuboid
1	*	1	$C_{t_1 \wedge t_3}$	2-D cuboid
*	1	1	$C_{t_2 \wedge t_3}$	2-D cuboid
1	1	1	$C_{t_1 \wedge t_2 \wedge t_3}$	3-D cuboid

or all of the d tags defining the cuboid, it is optimized to answer queries that include all d dimensions; if fewer are specified, several cell measures must be aggregated. For example, C_{t_1} (the centroid for all documents having tag t_1, regardless of whether or not they also have tags t_2 and t_3) can be computed as $C_{t_1 \wedge t_2 \wedge t_3} + C_{t_1 \wedge t_2 \wedge \neg t_3} + C_{t_1 \wedge \neg t_2 \wedge t_3} + C_{t_1 \wedge \neg t_2 \wedge \neg t_3}$, which requires accessing four of the 3-D cuboid's cells. In general, to answer a conjunctive query involving t tags by using a d-D cuboid defined over d tags with $d \geq t$, the set of tags defining the cuboid must include the set of tags used in the query and 2^{d-t} cells must be aggregated.

5.2.2 Individual Cell Materialization

Given a tag set $T = \{t_1, \ldots, t_d\}$, instead of materializing a whole d-D cuboid, this strategy materializes $I(T)$, the set of cells corresponding to all conjunctive queries without negation, which can be defined as follows:

$$I(T) = \{X_t.\mathit{alltags} \mid t \in (2^T - \emptyset)\} \tag{5.1}$$

where X_t is a cuboid for tag set t and *alltags* refers to the cell corresponding to all tags present. Table 5.1b shows the set of cells that will be materialized for $T = \{t_1, t_2, t_3\}$. The source column of the table identifies the cuboid from which the cell is taken.

For $|T| = d$, this approach materializes $2^d - 1$ cells, which is equal to the number of cells in the d-D cuboid. As shown in the last column of Table 5.1a, the set centroids for any cell in the 3-D cuboid for tag set $\{t_1, t_2, t_3\}$ can be derived using the set of cells in $I(\{t_1, t_2, t_3\})$. In general, similar conversions can be derived by taking advantage of the inclusion–exclusion principle.

Table 5.2: Cost of answering simple queries using a materialized t-D cuboid versus the individual cell materialization strategy

Query			**Cost**	
Length	**Pattern**	**Count**	**t-D cuboid**	$I(T)$
1	t_1	t	2^{t-1}	1
2	$t_1 \wedge t_2$	$\binom{t}{2}$	2^{t-2}	1
2	$t_1 \wedge \neg t_2$	$t(t-1)$	2^{t-2}	2
2	$t_1 \vee t_2$	$\binom{t}{2}$	$3 \times 2^{t-2}$	3

5.2.3 Query Performance Evaluation

For tag set T, full cuboid materialization and $I(T)$ both require the same number of cells to be stored, and both can answer all Boolean queries over T. However, the cost to answer a query depends on which set of materialized cells the query engine stores. Table 5.2 shows the number of cells that need to be aggregated to compute the answers to all queries involving one or two of t tags using a t-D cuboid vs. using individual cells included in the corresponding $I(T)$. In this table, the column labeled "count" shows how many distinct queries have the format shown in the column labeled "pattern"; for example, given four possible tags, there are 12 distinct queries that involve the conjunction of one (positive) tag and one negated tag; to answer any one of these queries, we need to access four cells in the cuboid (for every combination of tag presence and absence for the remaining tags), but only the two cells from $I(T)$ that correspond to the sets of documents having each tag.

Table 5.3 compares the cost of computing set centroids for all possible 3-tag queries when relying on a materialized 3-D cuboid against the cost when relying on cells materialized using $I(T)$. For three tags, there are seven nonoverlapping sets of documents (corresponding to the seven cells in the 3-D cuboid) and thus $2^7 - 1$ equivalence classes of queries. For each class, we found the minimum-length query (one with fewest literals) and, using these, tabulated the number of queries of each possible length against the cost (number of cells) needed to answer them under each materialization strategy. The 127 queries that were used to produce the table are listed in the Appendix. For very short queries, the query cost is lower when using $I(T)$ than when using the cuboid, and, importantly, the difference—as well as the length of query for which $I(T)$ outperforms the cuboid—becomes more pronounced as the number of dimensions in the materialized cuboids increases.

To compare query runtime when adopting the full cuboid materialization model against using the $I(T)$ model, we design a neutral query workload that has no bias toward any type of queries: each query includes at most three tags (with repetitions allowed), and

Table 5.3: Number of minimal queries having given costs and query lengths when using a 3-D cuboid (Cub) or $I(T)$ strategy

	Query length																			
	1		**2**		**3**		**4**		**5**		**6**		**7**		**8**		**9**		**10**	
Cost	Cub	I(T)	Cub	I(T)	Cub	I(T)	Cub	I(T)	Cub	I(T)	Cub	I(T)	Cub	I(T)	Cub	I(T)	Cub	I(T)	Cub	I(T)
1		3		3	7	1														
2			9	6		6			9		3									
3				3	15	12		9	15	3		3			5					
4	3					3	15	6	4	16	12	3				1			1	
5					9	6			9	9		6			3					
6			3			3			3	6	1	1				3				
7					1	1				6		3				4				1

the frequency of occurrence for each query depends only on its length and is independent of how many times it requires negation, union, intersection, or a combination of these operations. Using Table 5.3, Table 5.4 shows the average cost when using the $I(T)$ and complete cuboid (C) materialization strategies to answer queries when the query length probability distribution is uniform, zero truncated Poisson, and geometric. For all but the uniform distribution (where performance differs by only 10%), $I(T)$ materialization outperforms the full cuboid materialization approach, even when all queries involve fewer than four distinct tags.

Additionally, we evaluated the performance of the two materialization strategies with a query workload model derived from the analysis of the PubMed query log. Since it was observed that the 'NOT' operator occurred in only 1% of queries, the derived model will not generate queries that have that operator. This leaves us with 18 queries that it can generate, which are characterized by the number of 'AND' and 'OR' operators that they use. The probability of seeing queries with a 'AND' operators and o 'OR' operators is calculated from the probability distribution observed in Figure 1.3 under the assumption of independence of the two distributions. Table 5.4 shows that $I(T)$ materialization outperforms complete cuboid (C) materialization on the resulting generated workload.

Since we expect short queries to be more frequent than long ones, it is advantageous to use the individual cell materialization strategy.

5.2.4 Storage Performance Evaluation

The choice of materialization strategy affects the storage efficiency of the system, which can be evaluated by looking either at the amount of storage space necessary to support a fixed set of queries or at the number of queries that can be answered when a fixed amount of storage space is used. In this section, the storage efficiency of $I(T)$ materialization is

Table 5.4: Average cost of answering query

Prob distribution of tag count	**Average cost C**	**I(T)**
Uniform	3.7	4.1
Zero truncated Poisson (mean=1)	3.6	1.6
Zero truncated Poisson (mean=2)	3.5	2.3
Zero truncated Poisson (mean=3)	3.4	2.9
Geometric	3.6	1.9
Based on PubMed query log	3.0	1.3

compared to two partial materialization strategies that rely on full cuboid materialization: thin cube shells and shell fragments.

Thin Cube Shell

The thin cube shell approach, described in Section 2.5.2, relies on materializing all cells in all cuboids of a prescribed depth. As a result, the number of cuboids, and in turn the number of cells, that would need to be materialized grows rapidly as the number of dimensions increases. For a collection with T tags, thin cube shell materialization with d-dimensional cuboids requires $\binom{|T|}{d}(2^d - 1)$ cells to be materialized. In contrast, the $I(T)$ approach that supports all queries up to d tags requires $\sum_{i=1}^{d} \binom{|T|}{d}$ cells to be materialized.

For example, since the NYT uses 1,015 tags, there are $\binom{1,015}{3}$ three-dimensional cuboids, which corresponds to 1.2×10^9 cells that would need to be materialized by the thin cube shell strategy. On the other hand, the $I(T)$ approach requires 1.7×10^8 cells to be materialized, which is 7 times less. However, for most of those tag combinations there are at most 50 corresponding documents, and often there are none at all. Therefore, there is no need to materialize all cells with the $I(T)$ approach or all cuboids with the thin cube shell approach. With $I(T)$ materialization, unnecessary cells can be easily pruned, and only 38,368 cells are required to answer all queries that involve up to three distinct tags and contain more than 50 documents (which corresponds to the sum of the first three columns in Table 3.1) or 60,100 cells are required to ensure all the conjunctive queries that produce a result set above the threshold size can be answered efficiently. On the other hand, the thin cube shell approach requires 25,010 3-D *cuboids* to be stored, which corresponds to 175,070 cells (at 7 cells per cuboid). That is, the thin cube shell approach requires almost three times as many cells as the $I(T)$ approach if the cuboid size is chosen to be 3, and the multiplier gets larger as the prescribed cuboid size increases.

Shell Fragments

The shell fragment approach, proposed by Li et al. [Li et al., 2004] and described in Section 2.5.2, relies on materializing all cuboids for each fragment, where the fragments form a partitioning of the tags. For the NYT's 1,015 tags, we would need to store 338 fragments corresponding to tag triples and one fragment corresponding to the remaining tag, which would result in 6,423 cells being stored (at 19 cells per fragment for the triples). This is 11% of the size needed by the $I(T)$ storage approach that stores all 60,100 cells (and can answer all conjunction queries that produce result sets of size greater than 50), but that design is efficient only for queries including tags that are *all* found in the same fragment—at most 338 out of 20,905 tag triples that correspond to document sets larger than our threshold. When the tags specified in a query are not all found in the same fragment, the set of fragments that contain all the involved tags need to be intersected to identify which documents need to be aggregated online, and the response time might be unacceptable if the number of documents that need to be aggregated is above the threshold. If we choose to partition by six tags per fragment instead, we will require 111,496 cells (86% more than what is needed by $I(T)$), and still at most 170 out of 2,401 important sextets of tags will appear within a single fragment.

The effectiveness of answering conjunctive queries of various lengths on the NYT and ACM collections, when relying on shell fragment materialization with fragment sizes of six and three, are analyzed in Table 5.5. The fragments used are nonoverlapping and chosen using a greedy heuristic that builds fragments that can answer the longest conjunctions. In both the NYT and ACM collections only a very small percentage of conjunctions can be answered using the nonoverlapping shell fragments. Thus, the partial materialization generated by shell fragments cannot guarantee acceptable performance when tags co-occur with many other tags and there is a high order of tag co-occurrences, as is true in both the NYT and ACM collections.

5.3 Partial Materialization Strategies

Because each centroid term vector includes a (sum, count) pair for each of the m most significant terms found in the document collection (Section 1.1.1), and for our collections $m = 500$, the space for storing a single centroid can be as much as 4KB (if uncompressed).

Table 5.5: Analysis of the number of conjunctive queries producing result sets above the threshold size that can be answered when 6-D or 3-D shell fragment materialization is used on **a)** NYT **b)** ACM

NYT

Query	Unique	**6-D shell fragment**		**3-D shell fragment**	
length	conjunctions	Count	%	Count	%
1	1,015	1,015	100.00	1,015	100.00
2	16,448	1,002	6.09	564	3.43
3	20,905	585	2.80	119	0.57
4	12,217	280	2.29	0	0.00
5	5,289	90	1.70	0	0.00
6	2,401	13	0.54	0	0.00
7	1,152	0	0.00	0	0.00
8	493	0	0.00	0	0.00
9	151	0	0.00	0	0.00
10	27	0	0.00	0	0.00
11	2	0	0.00	0	0.00
Total	60,100	2,985	4.97	1,698	2.83

ACM

Query	Unique	**6-D shell fragment**		**3-D shell fragment**	
length	conjunctions	Count	%	Count	%
1	9,098	9,098	100.00	9,098	100.00
2	14,262	2,006	14.07	1,199	8.41
3	5,280	797	15.09	186	3.52
4	3,860	350	9.07	0	0.00
5	3,700	114	3.08	0	0.00
6	3,199	17	0.53	0	0.00
7	2,390	0	0.00	0	0.00
8	1,520	0	0.00	0	0.00
9	776	0	0.00	0	0.00
10	297	0	0.00	0	0.00
11	79	0	0.00	0	0.00
12	13	0	0.00	0	0.00
Total	44,474	12,382	27.84	10,483	23.57

Even if centroids were compressed, they will still require considerable space. Therefore, it is worthwhile to avoid materializing cells as much as feasible.

To this end, three partial materialization strategies are proposed: threshold materialization (TM), threshold materialization with ancestors (TMA), and materialization of cluster centroids (MCC). For each materialization strategy, we give algorithms for choosing the centroids to materialize and for answering queries using those centroids with appropriate compensations when a requested cell centroid is not materialized.

5.3.1 Threshold Materialization

Assuming that we can afford to access and aggregate at most k documents when computing a centroid (Section 1.1.4), we start by precomputing and materializing the centroids for all conjunctive queries for which the result contains at least k documents. We therefore need to identify which combinations of tags produce "cells of significant size" after intersection, as enumerated for Table 3.1. Algorithm 3 returns a list M of intersection cells that have more than k member documents.

Algorithm 3 TM: Find cells exceeding threshold filter

```
Input: Threshold $k$; Tags $T = \{(t_i,\ t_i \rightarrow S_{\{t_i\}})\}$
Returns: Set of (candidate) cells with their centroids
$M \leftarrow \emptyset$, $u \leftarrow \emptyset$
$L \leftarrow u.augmentSet(T)$                ▷ start with a list of tag sets of size 1
while $|L| > 0$ do
    $u \leftarrow L.dequeue()$
    if $|S_u| > k$ then                       ▷ include augmented tag sets that represent more than $k$ documents
        $M \leftarrow M \cup \{(u, C_u)\}$
        $L.enqueue(u.augmentSet(T))$          ▷ ... and continue to check supersets of $u$
    end if
end while
return $M$
```

The algorithm is based on the simple observation that including additional tags in a conjunctive query cannot increase the number of documents in the resulting intersection. It starts with all possible single tags, which correspond to 1-D cells, and the set of documents associated with each tag. Using the method *augmentSet*(), it then repeatedly includes one more tag in the conjunction. (The method returns a list of sets, each augmenting the base set with one tag not already included in that base. To avoid repeated consideration, only tags that have a higher index than the maximum tag index in set u are included in the list of sets returned by $u.augmentSet(T)$.) The tag sets (together with their corresponding document sets) that have more than k documents and therefore require further exploration are kept in a queue L. Each time the number of documents in a cell exceeds the threshold k, it is included in the result set, the intersections with each remaining tag is computed, and the resulting augmented tag sets are enqueued on L for further consideration. The algorithm continues to examine cells with more and more intersecting tags until no further candidates have more than k documents.

Given this partial materialization, the following steps are performed to evaluate a query:

1. Transform the query into an equivalent representation R using the inclusion–exclusion principle.
2. For each resulting conjunction, check if the corresponding cell has been materialized and, if so, retrieve the centroid measure.
3. For each of the nonmaterialized conjunctions:
   (a) determine the set of documents in the intersection (merge the tags' postings lists);
   (b) retrieve and aggregate the document term vectors to generate the corresponding centroid measure.
4. Combine the centroid measures in accordance with R.

5.3.2 Threshold Materialization with Ancestors

Table 3.2 shows that the set of documents found by intersecting a set of tags G is often equal to or very similar to the set found by intersecting tags in $G' \subset G$. With this insight, we extend Algorithm 3 to include the additional materialization constraint that the size of the symmetric difference between each cell and its closest materialized ancestor must be greater than k. This materialization approach, described by Algorithm 4, is designed to take advantage of the similarity between cells that involve similar tags. (The method $M.getClosestAncestor(u)$ retrieves the closest materialized ancestor for u as measured by symmetric difference.) Notice, however, that even if a cell is not materialized because it is similar to a materialized ancestor, some of its descendant cells might still require materialization; in this respect, the approach uses a greedy algorithm rather than finding the optimal set of cells to materialize. Nevertheless, for collections with a large amount of co-occurrence between tags, this approach will provide significant storage savings, and for collections with very little similarity, it will perform like the TM algorithm.

An additional lookup table is stored for this strategy, where for each nonmaterialized cell c having more than k documents, we store a pointer to the closest materialized ancestor a. Although this requires a small amount of space, it is far less than what is required to store a centroid.

Algorithm 4 TMA: Find cells exceeding threshold filter given materialized ancestors

```
Input: Threshold $k$; Tags $T = \{(t_i,\ t_i \rightarrow S_{\{t_i\}})\}$
Returns: Set of cells with their centroids
$M \leftarrow \emptyset$, $u \leftarrow \emptyset$
$L \leftarrow u.augmentSet(T)$
while $|L| > 0$ do
    $u \leftarrow L.dequeue()$
    if $|S_u| > k$ then                          ▷ centroid might need to be materialized
        $a \leftarrow M.getClosestAncestor(u)$
        if $|S_a \triangle S_u| > k$ then        ▷ ... but not if the document set is sufficiently close to an ancestor
            $M \leftarrow M \cup \{(u, C_u)\}$
        else
            $M \leftarrow M \cup \{(u, *a)\}$    ▷ ... (in which case, just point at that ancestor)
        end if
        $L.enqueue(u.augmentSet(T))$
    end if
end while
return $M$
```

The following steps are now required to evaluate a query:

1. Transform the query into an equivalent representation R using the inclusion–exclusion principle.

2. For each resulting conjunction, check if it has been materialized and, if so, retrieve the centroid measure.

3. For each of the nonmaterialized conjunctions:

    (a) determine S_c, the set of documents in the intersection (merge postings lists for the given tags);

   (b) if $|S_c| \leq k$, retrieve and aggregate the document term vectors to generate the corresponding centroid measure.
   (c) if $|S_c| > k$:
      i. retrieve document member set S_{*a};
      ii. retrieve and aggregate the document term vectors in set $S_{*a} - S_c$ and call the result δC;
      iii. compute the centroid measure to be the value $C_{*a} - \delta C$.

4. Combine the centroid measures in accordance with R.

5.3.3 Materialization of Cluster Centroids

A third approach is to compute centroids for carefully selected document sets that do not necessarily correspond to cells in the data cube. Instead of depending on the closest materialized ancestor to provide an approximate centroid, it stores centroid measures of sets that do not correspond to any specific query but from which a result to a query can be derived. For document collections with little similarity among cells, this algorithm will materialize at most as many cells as Algorithm 3.

Algorithm 5 starts by calling the *candidateCells*() function (returning the sets of documents for the cells chosen to be materialized by Algorithm 3) to obtain the set M of document sets representing cells whose centroids cannot be computed by merely combining at most k document term vectors. Next, the method *closePairs*() initializes a priority queue Q that will contain (c_i, c_j, δ) triples, ordered by ascending δ, where $c_i, c_j \in M \wedge c_i \neq c_j$ and $\delta = |S_{c_i} \triangle S_{c_j}|$. In this method, all child and sibling relationships among pairs of cells are examined to identify those pairs (c_i, c_j) that have a low δ. This corresponds to initializing Q with the following candidate pairs: $c_i \subset c_j \wedge |c_j| - |c_i| = 1$ (parent–child relationship) or $|c_i| = |c_j| \wedge |c_i \triangle c_j| = 2$ (sibling relationship).

The algorithm then applies complete-link clustering [Manning et al., 2008] to find collections of highly overlapping document sets. Unlike traditional hierarchical clustering, however, we do not care about the order in which clusters are merged, as long as all the members of each cluster satisfy the complete linkage requirement that they are within a symmetric distance of $2k$ from the furthest member in the cluster (which ensures that all the sets' centroids can be computed from the cluster centroid by considering at most k documents). This relaxation in preserving the cluster hierarchy allows an efficient implementation for large numbers of cells by using a standard union-find algorithm [Sedgewick and Wayne, 2011]. To accomplish its clustering, the algorithm first invokes the method

Algorithm 5 MCC: Find clusters for materialization

```
Input: Threshold $k$; Tags $T = \{(t_i,\ t_i \rightarrow S_{\{t_i\}})\}$
Returns: Set of clusters with their centroids
$M \leftarrow candidateCells(k, T)$        ▷ use Algorithm 3 to identify all sets requiring materialization
$Q \leftarrow M.closePairs()$        ▷ collect pairs of sets with small symmetric distance
$G.initializeClusters(M)$        ▷ every candidate cell is initially in its own cluster
while $|Q| > 0$ do
    $q \leftarrow Q.dequeue()$        ▷ greedily merge clusters using union-find
    $p_1 \leftarrow G.getCluster(q.c1)$
    $p_2 \leftarrow G.getCluster(q.c2)$
    if $p_1 \neq p_2$ then
        $p_u \leftarrow p_1 \cup p_2$
        if $p_u.maxDistance() \leq 2k$ then        ▷ if all pairs are within $2k$, merge clusters
            $G[q.c1].setCluster(p_u)$
            $G[q.c2].setCluster(p_u)$
            $M \leftarrow M - \{p_1\} - \{p_2\}$
            $M \leftarrow M \cup \{p_u\}$
        end if
    end if
end while
return $M$
```

initializeClusters() to generate a disjoint set data structure G, where $G[M_i] = M_i$, that is used to track which cells (document sets) are assigned to which clusters (partitions). The method *getCluster*() retrieves the partition to which a specified cell belongs, *setCluster*() assigns a specified cell to a partition, and *p.maxDistance* () returns $\max\limits_{c_i,c_j \in C}(|S_{c_i} \triangle S_{c_j}|)$.

Algorithm 5 returns a set of partitions $p_1, \ldots, p_n$, where each partition p_i represents a cluster of cells $S_{(i,1)}, \ldots, S_{(i,i_n)}$ and each $S_{(i,j)}$ corresponds to a conjunction of tags. For each partition p_i, we determine a set of documents S_{p_i}, that is, within distance k of each $S_{(i,j)}$ (which must exist since no two documents in the partition are further than $2k$ apart). The term centroid C_{p_i} for this "artificial cell" is then calculated by aggregating together all the document term vectors found in S_{p_i}.

This strategy requires all the C_{p_i} measures to be materialized and the required cell centroids to be computed based on the closest materialized artificial cell. To accomplish this, we store a table with four attributes: *cell, cluster, docsToAdd, docsToRemove*; where *cell* corresponds to a cell representing a conjunction of tags, *cluster* is the centroid for the partition to which that cell is assigned, *docsToAdd* is the set of document IDs in the cell but missing when computing the partition centroid, and *docsToRemove* is the set of document IDs included when computing the partition's centroid but missing from the cell. The documents in *docsToAdd* and *docsToRemove* need to be retrieved and aggregated to the partition's centroid measure to determine the centroid for the cell. By construction, $|\mathit{docsToAdd}| + |\mathit{docsToRemove}| \leq k$.

Thus, at query time the following steps are performed to evaluate a query:

1. Transform the query into the equivalent representation R using the inclusion–exclusion principle.

2. For each resulting conjunction j, find the cluster centroid measure C_{p_j} from the cluster reference table.

3. If there is no match, retrieve and aggregate the document term vectors to generate the corresponding centroid measure.

4. Otherwise, retrieve the documents included in the *docsToAdd* and *docsToRemove* attributes and aggregate them with C_{p_j}.

5. Combine the centroid measures in accordance with R.

Table 5.6: Tag assignment to documents.

	d_1	d_2	d_3	d_4	d_5	d_6
t_1	✓	✓	✓	✓	✓	✓
t_2	✓	✓	✓	✓	✓	
t_3	✓	✓	✓			✓

Table 5.7: Materialized cells for TM, TMA, and MCC materialization strategies.

	TM	**TMA**	**MCC**
$S_1 = \{t_1\}$	$\{d_1, d_2, d_3, d_4, d_5, d_6\}$	$\{d_1, d_2, d_3, d_4, d_5, d_6\}$	
$S_2 = \{t_2\}$	$\{d_1, d_2, d_3, d_4, d_5\}$	$\{d_1, d_2, d_3, d_4, d_5\}$	
$S_3 = \{t_3\}$	$\{d_1, d_2, d_3, d_6\}$	$\{d_1, d_2, d_3, d_6\}$	
$S_4 = \{t_1, t_2\}$	$\{d_1, d_2, d_3, d_4, d_5\}$		
$S_5 = \{t_1, t_3\}$	$\{d_1, d_2, d_3, d_6\}$		
$S_6 = \{t_2, t_3\}$	$\{d_1, d_2, d_3\}$		
$S_7 = \{t_1, t_2, t_3\}$	$\{d_1, d_2, d_3\}$		
S_a			$\{d_1, d_2, d_3, d_4\}$

5.3.4 Comparative Example of Materialization Strategies

In this section, we demonstrate how the use of different materialization strategies affects which sets of documents have their centroids materialized, and how these materialized centroids are used to answer queries over tags. For simplicity, we assume a document collection that consists of six documents: $d_1, \ldots, d_6$; a set of 3 tags: t_1, t_2, t_3, whose assignment to documents is specified in Table 5.6; and materialization threshold $k = 2$.

Table 5.7 shows the document sets for which centroids are stored for each of the three materialization strategies. The TM approach materializes centroids for all document sets of size greater than 2 after considering every conjunction of tags, which results in materializing centroids for seven document sets. With TMA, only three document set centroids are materialized (corresponding to document sets for single tag queries: t_1, t_2, and t_3), since the other conjunctions are descendants of them and the size of the symmetric difference between them and the materialized sets is no more than the threshold 2. With MCC only a single centroid is materialized (for a set of documents that correspond to no tag conjunction), from which the centroids of all the conjunctive queries can be derived with no more than 2 DTVs involved.

Table 5.8 demonstrates how the materialized centroids produced by each of the three materialization strategies are used to derive a centroid for various conjunctive queries over

Table 5.8: Answering conjunctive queries with TM, TMA, and MCC materialization strategies.

Query	**TM**	**TMA**	**MCC**
t_1	C_1	C_1	$C_a + d_5 + d_6$
t_2	C_2	C_2	$C_a + d_5$
t_3	C_3	C_3	$C_a - d_4 + d_6$
$t_1 \wedge t_2$	C_4	C_2	$C_a + d_5$
$t_1 \wedge t_3$	C_5	C_3	$C_a - d_4 + d_6$
$t_2 \wedge t_3$	C_6	$C_3 - d_6$	$C_a - d_4$
$t_1 \wedge t_2 \wedge t_3$	C_7	$C_3 - d_6$	$C_a - d_4$

tags. Since with the TM strategy the materialized centroids match the document sets requested by the queries, no further work needs to be performed to produce the required results. When relying on the TMA and MCC strategies, the centroid for a query answer is derived by aggregating the centroid of the closest materialized document set with DTVs from individual documents.

5.4 Performance of Partial Materialization

Our approach to partial materialization was designed to perform well for the tagging patterns we observed in the NYT and ACM. In this section, we show that it indeed does well for those collections, as well as for the much larger PubMed collection[1]. The PubMed corpus consists of 244,553,378 MeSH terms (which we treat as tags) that are assigned to 20,997,401 documents, corresponding to 11.65 tags per document on average. The corpus identifies 71,690,729 assigned tags as "major," that is, the topics play a major part in the associated paper, yielding on average 3.28 major tags per document. PubMed's average major tag count per document is between the NYT and ACM; the number of documents and total number of tags are larger than the NYT, even when considering only the major tags; and the depth of tag co-occurrence is shallower than for either of the other two collections (compare Table 5.9 to Table 3.1).

In the remainder of this section, the performance of the three partial materialization strategies (TM, TMA, and MCC, which rely on the proposed $I(T)$ materialization defined in Section 5.2.2) are compared against six other materialization strategies. Specifically, we compare them against the shell fragment materialization (SF) with both 3-D and 6-D

[1]Available at http://mbr.nlm.nih.gov/Download/2014/Data/Full_MH_SH_items.gz.

Table 5.9: Number of conjunctions of n tags that contribute to high multi-way co-occurrence for PubMed, with threshold limit of 50.

n	1	2	3	4	5	6	7	8	9	10	11	12	Total
PubMed	114,946	218,596	38,392	7,410	1,305	221	43	5					380,918

cuboids[2], thin cube shell materialization (CS) with both 3-D and 6-D cuboids, materialization of hierarchical clusters used by Scatter/Gather (SG), and a standard IR approach, which is equivalent to the strategy with no materialization (NM).

Shell fragments are nonoverlapping, and for the evaluation they are chosen using a greedy heuristic that builds fragments that can answer the longest conjunctions. Similarly, thin cube shells were generated by a greedy algorithm tuned to minimize the number of stored cuboids while ensuring that all conjunctions of less than or equal to a fixed length can be answered from materialized cuboids. For both the shell fragment and thin cube approaches, each d-dimensional cuboid records its d dimensions and a centroid measure. In addition, both approaches also require a mechanism to map tags to the corresponding dimension column of the cuboid. If there is no cuboid that contains all the tags found in a query, that query is answered by accessing the postings list of documents with each tag (as is true when no appropriate centroid is materialized in using any of the strategies).

The LAIR2 [Ke et al., 2009] implementation of Scatter/Gather was designed to support a browsing interface in which exploration of the collection starts from a root set of document clusters and then follows various paths offered by the stored cluster hierarchy. To support this interface, it is necessary to store the hierarchy, as well as the centroid, representative documents, and most frequent terms for each cluster that corresponds to a node in the hierarchy. This infrastructure, however, does not provide support for efficient computation of a centroid for a set of documents that results from a conjunctive tag query, since it is extremely unlikely that the query result set matches any of the stored document sets exactly. As a result, Scatter/Gather must revert to online aggregation from the base documents like standard IR approaches.

The performance of each of the nine strategies is evaluated in terms of the amount of storage space it consumes and the execution cost of answering queries. Precision is not a consideration, since all matches are exact, as required by our problem definition.

[2]For the ACM-Model collection the longest conjunctive queries are of length 5 and so instead of materializing shell fragments and shell cuboids of size 6, we materialize cuboids of size 5 so that they are not penalized by storing unnecessary data.

5.4.1 Storage Cost

Table 5.10 shows the storage space consumed by each of the materialization strategies. Every strategy, including NM (no materialization), requires postings lists to map from tags to sets of documents associated with those tags, in order to answer queries that must rely on the base data (or to compensate for document sets that do not match the stored centroids exactly when using the $I(T)$ strategies). Thus the first column reflects the space for the postings lists alone. The remaining columns show the additional space used by each algorithm to store the materialized cells and other required supporting data when a centroid is represented by 500 terms. Figure 5.2 depicts the storage requirement as a multiplier with respect to the space used by TM.

Table 5.10: Storage cost comparison for materialization strategies (megabytes)

	NM	**TM**	**TMA**	**MCC**	**CS3**	**CS6**	**SF3**	**SF6**	**SG (range)**
NYT	18	248	129	104	686	2,252	42	447	[130, 5,603]
ACM	2	172	97	81	401	2,017	222	3,849	[52, 255]
PubMed	462	1,918	1,812	1,734	6,143	20,644	3,246	49,106	[2,051, 79,888]
NYT-Model	2	171	135	114	674	1,948	20	317	[114, 560]
ACM-Model	2	92	90	77	422	*860	120	*790	[53, 255]
S1	2	280	258	223	1,293	3,251	17	261	[114, 561]
S2	1	91	80	68	400	1,047	38	649	[52, 255]
S3	2	145	123	101	631	1,611	40	672	[52, 255]
S4	2	1,245	775	525	3,206	18,475	45	746	[53, 256]
S5	2	517	390	301	1,655	8,813	42	708	[53, 255]
S6	1	117	90	71	464	1,115	26	440	[52, 254]

* 5-D cuboids used for the ACM-Model for CS6 and SF6

Recall that, in addition to the cluster centroids, MCC must store a cluster reference table with as many as k document IDs for every cell that would be materialized by TM. In practice, many fewer IDs need to be stored: The mean number of documents in the difference between an unmaterialized cell and the corresponding cluster centroid is 8.1 for the NYT, 0.75 for the ACM, and 4.6 for PubMed; for the NYT-Model it is 0.9, and it is 0.9 for the ACM-Model; it is similarly low for the other six synthetic collections. As a result, even with the additional cost of storing the cluster reference table, MCC requires less space than either TM or TMA.

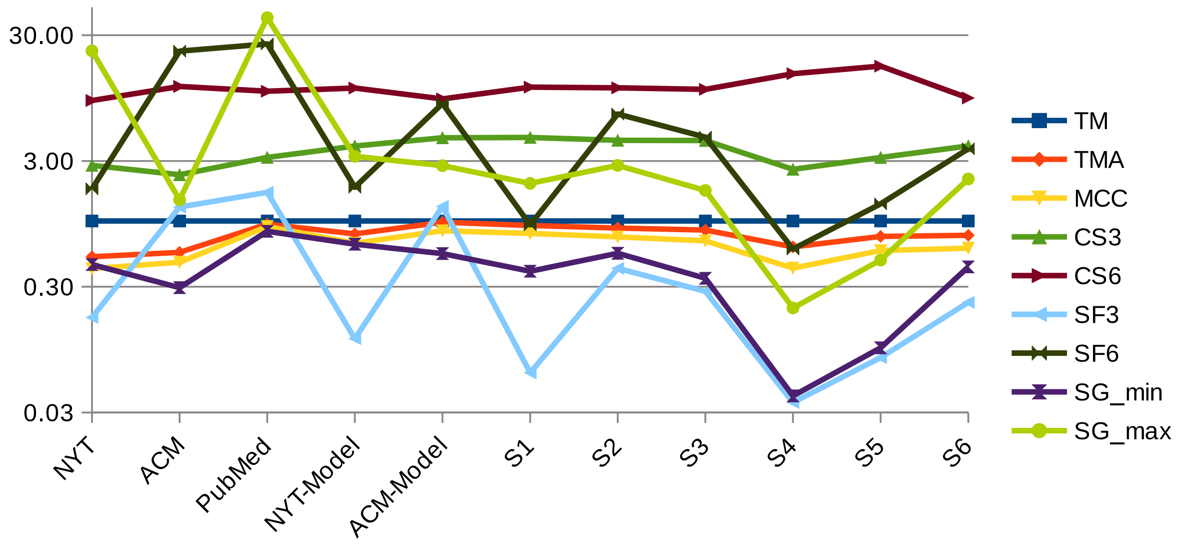


Figure 5.2: Space multiplier relative to TM.

The shell fragment strategy using 3-D cuboids consumes less space than other partial materialization strategies for all collections other than the ACM, ACM-Model, and PubMed, but it exhibits very poor query time performance, as shown in Section 5.4.2.

In order to save clustering time, Scatter/Gather requires space that is proportional to the collection size but independent of the tag patterns, so it performs relatively well on small collections with rich tag structures, such as S4. To provide a fair comparison to our approaches, we applied a similar partial materialization strategy for storing the clusters in the hierarchy: Only nodes exceeding the predefined size threshold k for each collection are materialized. As a result, the number of nodes that need to be materialized for Scatter/Gather depends on the form of the dendogram representing the cluster hierarchy: If k individual documents are always clustered before any two clusters are merged, $\frac{N}{k}$ clusters would need to be stored, but in the worst case, the dendogram is essentially linear and $N - k$ clusters might need to be materialized. The corresponding ranges of space that bracket the actual space that would be required by Scatter/Gather are shown in the final column of the Table 5.10. Scatter/Gather does not scale well to very large collections such as the NYT and PubMed, where for the NYT (*at least* 25%) more space is required and for PubMed (*at least* 18%) more space is required than when using MCC. For the ACM collection the partially materialized cluster hierarchy for LAIR2 might be competitive in space with MCC, but when we performed hierarchical clustering on the documents in the

ACM collection using single linkage and cosine similarity, we found that 208MB would be required; this is closer to the upper bound of the range than to the lower bound and 2.6 times as much space as required by MCC. We hypothesize that for the other collections the actual storage space used for LAIR2 would be similarly much higher than the theoretical lower bound.

In summary, the TM, TMA, and MCC approaches, which rely on individual cell materialization $I(T)$, typically require fewer cells to be stored, and thus less storage space, than alternative approaches, and among these three, the MCC approach is the most storage efficient.

5.4.2 Query Execution Cost

Table 5.11 and Figure 5.3 show the number of cell accesses required to answer all possible conjunctive queries having a result that contains more than k documents. The TM and Scatter/Gather strategies are not included in the table. TM precalculates and materializes all answers to such queries, and hence its cost is universally 1 (with standard deviation 0). The Scatter/Gather approach, on the other hand, has similar query time performance for retrieving centroids of nodes stored in its cluster hierarchy, as would be expected from a fully materialized solution; however, it was not designed to calculate centroids for document sets that do not map to any of the nodes found in the hierarchy. Therefore, for almost all conjunctive queries Scatter/Gather does not have the necessary infrastructure to compute centroids from its materialized data, and therefore, it must resort to the approach with no materialization. As a result the mean number of cells accessed per query across all queries for SG is just marginally less than the figures shown in the NM column of the table.

Since the query time is proportional to the average number of cells that need to be aggregated, these results represent comparative run times when each centroid is equally likely to be requested. By design, all three materialization strategies based on individual cell materialization (TM, TMA, MCC) have a computation cost that is less than the collection-specific computation threshold k and less than when using thin cube shells or shell fragments. On the other hand, when no cells are materialized (NM)—when IR is used alone—or when using Scatter/Gather with or without prematerialization, the computational costs far exceed acceptable response times.

Thus, summarizing space and time, all three $I(T)$ approaches have excellent query time performance (low mean and small standard deviation) while consuming generally the least amount of storage compared to the other materialization techniques. Furthermore, the three options provide a good space–time tradeoff, with TM being the largest and fastest

Table 5.11: Mean number of cells aggregated per query (with standard deviations σ)

Collection	# Queries	TMA	MCC	CS3	CS6	SF3	SF6	NM
NYT	60,100	$4.9_{\sigma=9}$	$9.1_{\sigma=14}$	$38_{\sigma=88}$	$11_{\sigma=11}$	$162_{\sigma=533}$	$152_{\sigma=432}$	$238_{\sigma=1461}$
ACM	44,474	$1.4_{\sigma=1}$	$1.8_{\sigma=1}$	$3.6_{\sigma=2}$	$14_{\sigma=11}$	$5.9_{\sigma=6}$	$5.5_{\sigma=6}$	$12_{\sigma=37}$
PubMed	380,918	$2.3_{\sigma=6}$	$5.6_{\sigma=12}$	$4.5_{\sigma=17}$	$20_{\sigma=9}$	$73_{\sigma=123}$	$67_{\sigma=114}$	$265_{\sigma=964}$
NYT-Model	44,337	$1.4_{\sigma=1}$	$1.9_{\sigma=1}$	$3.4_{\sigma=9}$	$10.3_{\sigma=6}$	$17.2_{\sigma=47}$	$15.2_{\sigma=33}$	$27.7_{\sigma=221}$
ACM-Model	23,621	$1.1_{\sigma=0}$	$1.9_{\sigma=2}$	$2.3_{\sigma=1}$	*$9.1_{\sigma=4}$	$7.4_{\sigma=8}$	*$6.9_{\sigma=7}$	$19_{\sigma=49}$
S1	72,684	$1.2_{\sigma=1}$	$1.9_{\sigma=1}$	$2.2_{\sigma=3}$	$10.9_{\sigma=5}$	$17_{\sigma=38}$	$15.2_{\sigma=29}$	$25.5_{\sigma=163}$
S2	23,651	$1.3_{\sigma=1}$	$2.0_{\sigma=2}$	$2.3_{\sigma=2}$	$14_{\sigma=6}$	$11_{\sigma=16}$	$8.7_{\sigma=12}$	$23_{\sigma=75}$
S3	37,440	$1.3_{\sigma=1}$	$2.1_{\sigma=2}$	$2.4_{\sigma=3}$	$13_{\sigma=6}$	$11_{\sigma=17}$	$9.2_{\sigma=13}$	$20_{\sigma=69}$
S4	325,340	$1.8_{\sigma=1}$	$2.5_{\sigma=1}$	$5.5_{\sigma=7}$	$7.0_{\sigma=5}$	$13_{\sigma=23}$	$12_{\sigma=19}$	$14_{\sigma=44}$
S5	134,765	$1.5_{\sigma=1}$	$2.3_{\sigma=2}$	$4.3_{\sigma=5}$	$8.1_{\sigma=5}$	$13_{\sigma=21}$	$12_{\sigma=19}$	$16_{\sigma=51}$
S6	30,441	$1.4_{\sigma=1}$	$2.1_{\sigma=1}$	$2.7_{\sigma=4}$	$13_{\sigma=6}$	$12_{\sigma=19}$	$8.7_{\sigma=12}$	$22_{\sigma=83}$

* 5-D cuboids used for the ACM-Model for both CS6 and SF6

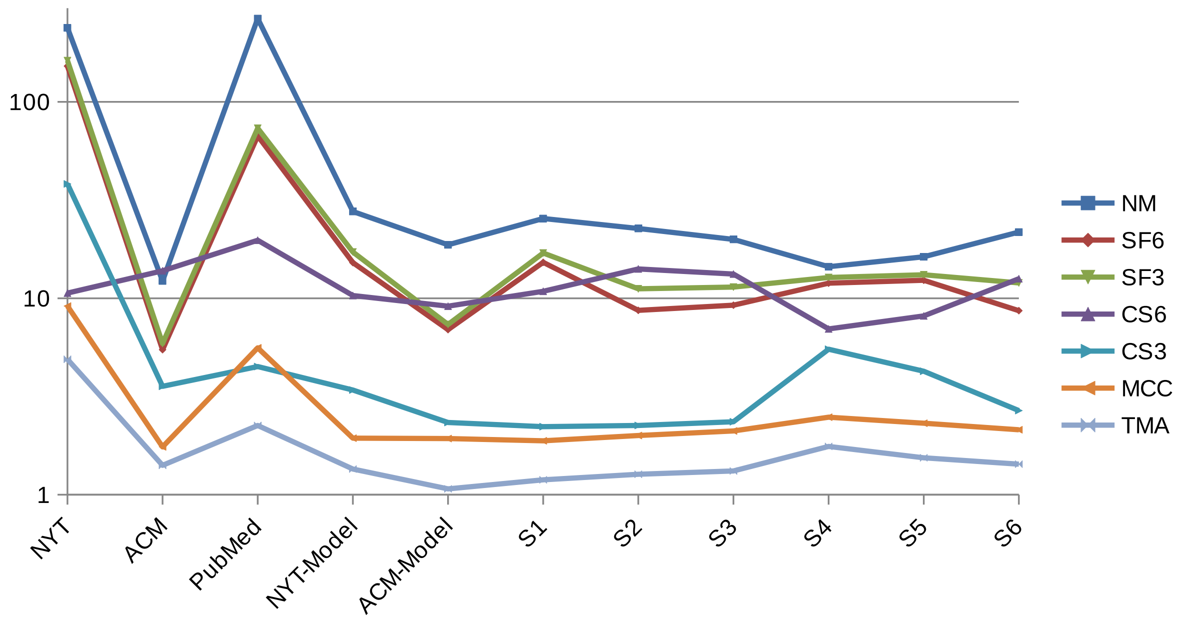


Figure 5.3: Mean number of cells accessed per query.

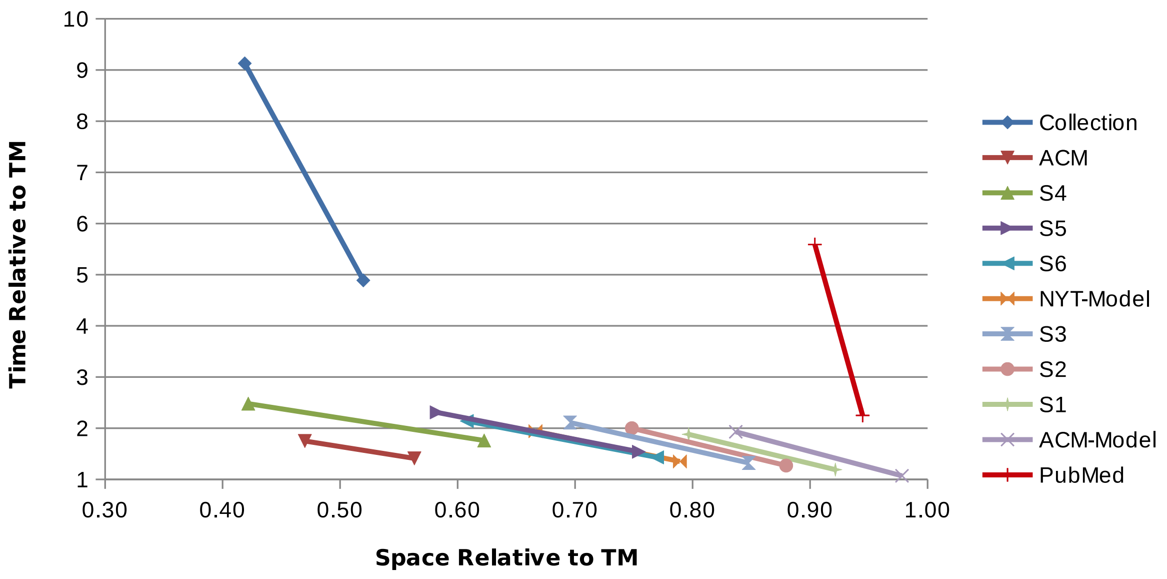


Figure 5.4: Space–time tradeoff for $I(T)$ strategies, where all points are relative to TM. (for each collection, the upper left point represents MCC and the lower right point represents TMA)

and MCC being the smallest but slowest for all 11 data collections (see Figure 5.4, where TM resides at the lower right-hand corner of the graph—at (1.00,1)—for each collection). Scatter/Gather can be fast, but only if it consumes an unacceptable amount of space for large collections and only if the query workload is severely restricted to correspond to the sets of documents that happen to be chosen by the clustering algorithm. Shell fragments with 3-D cuboids can be space-efficient, but only at the expense of unacceptably slow execution.

5.5 Conclusions

The focus of this chapter was on developing a partial materialization infrastructure that supports fast calculation of centroids for sets of documents defined through Boolean queries over tags in large multi-tagged document collections. We developed a novel materialization strategy that stores data at the granularity of individual cells instead of full cuboids, the traditional approach used by competing high dimensional OLAP systems. We have shown that by storing data at higher granularity we can achieve a lower storage cost than is possible with full cuboid materialization, while at the same time providing a lower query execution cost for expected query workloads. Furthermore, we developed three partial materialization strategies that work on top of the individual cell materialization and provide further saving in space. The proposed partial materialization strategies were compared against competing approaches on a set of three real and eight synthetic collections and were shown to provide a significant savings in storage and query execution costs.

In this chapter we focused on efficient storage and calculation of the set centroid measure, which is essential input for deriving a medoid, a larger set of indicative documents, or a set of indicative terms. The next step is to investigate what is the appropriate vocabulary that should be stored in each of the centroids such that the derived measures will be of high quality. After that we will show how these partial materialization strategies can be used effectively in a faceted browsing system.

Chapter 6

Dealing with a Diverse Vocabulary

The choice of term vocabulary stored as part of materialization has an impact on both the amount of storage space that a materialization plan will require and on the contents of the summary measures produced. In particular, the storage cost of a materialization plan is directly related to the number of terms used to represent a centroid of a document set. In the evaluation of the various partial materialization strategies in Chapter 5, the number of terms stored per centroid was fixed arbitrarily at 500 terms, globally defined for the whole collection and common to centroids summarizing each of the document sets. However, using locally-defined vocabularies might improve the computation of centroids and yield better collections of indicative documents and indicative terms (see Section 2.4.6).

It is important for materialized centroids to contain good coverage of the top k terms relevant to the specific document set, since the two text centric summary measures that we are supporting (sub-collection indicative terms and indicative documents) are derived directly from the stored centroids. The choice of centroid vocabulary has a direct effect on the indicative terms produced and the set of indicative documents chosen. If the key terms to a specific document set are missing from the centroid, then the quality of the indicative terms will be affected since more generic terms will be used as indicative terms instead, which may be less meaningful and informative. In addition, the documents chosen as representatives may not be the ones that represent the key concepts to the set. For this reason it is important for the centroids to store many set-specific terms for each possible set of documents.

Since the total available global vocabulary for the NYT is 238,065 terms and for the ACM is 13,015, as indicated in Table 3.3, and the current approach only uses 500 terms out of that vocabulary to describe the centroids, it is questionable whether this vocabulary is rich enough to effectively summarize the sets of documents in the collection. If for each collection the global vocabulary was instead determined by taking the *single* most significant term from each of the 1,015 tags found in the NYT and 9,098 tags found in the ACM, then, after removing duplicates, a global dictionary consisting of 867 terms would be required for the NYT and 3,635 terms would be needed for the ACM. However, to obtain a better coverage of concepts, more than one term would likely need to be contributed by each tag to the global vocabulary. A more reasonable global vocabulary would be formed if each tag contributed its top 5 terms to the global vocabulary, which would result in a global vocabulary of 2,479 for the NYT and 10,318 for the ACM. The relationship between the size of global vocabulary and the number of top k terms contributed by each tag was shown in Table 3.3.

However, having a larger global vocabulary can significantly increase the storage cost of the whole materialization, as can be observed in Table 6.1, where increasing the global vocabulary in the NYT from 867 to 2,479 (taking the top single feature from each tag vs. taking the top five features from each tag) results in an increase of the storage cost of the partial materialization by a factor of 2.5. Increasing the global vocabulary for the ACM collection from 3,635 to 10,318 (taking the top single feature from each tag vs. taking the top five features from each tag) results in an increase of the storage cost of the partial materialization by a factor of 1.6.

Not only is increasing the global vocabulary costly, but the earlier analysis of the term occurrence distribution in multiple tags in Figure 3.5 shows that most tag informative terms occur in few tags and the global vocabulary has poor containment of the top k tag specific terms, as shown in Figure 3.6 for the NYT and Figure 3.7 for the ACM: In the case of the NYT, on average only 49 out of 100 terms were covered by the global vocabulary, and for the ACM 24 out of 50 terms were contained. When taking the top 500 terms for each of the tags in the NYT or ACM collection, on average only 162 terms are covered by the global vocabulary in the NYT and 181 terms by the global vocabulary in the ACM. The observed locality of terms and mediocre coverage of the top k tag specific terms raises questions about the suitability of using a global vocabulary to support document collections that cover many topics with topic-specific vocabulary.

Thus to ensure that the majority of the top k terms are available for each tag (to provide good candidate indicative terms for single tag queries), a large global vocabulary is required, which will consume a large amount of storage. To ensure that the global vocabulary contains indicative terms for sets of documents corresponding to a multi-tag query, even more global terms may need to be stored.

Table 6.1: Storage cost (bold entries reflect parameters that yield similar storage costs for the three approaches)

Storage approach	**NYT**	**ACM**
No Filtering	171,727,281	10,441,546
Global with 5 terms from each tag	**28,083,167**	7,420,447
Global with 2 terms from each tag	17,168,627	5,934,915
Global with 1 terms from each tag	11,062,033	**4,752,044**
Local with 800 terms from each tag	28,174,351	7,394,189
Local with 796 terms from each tag	**28,065,827**	
Local with 500 terms from each tag	19,387,494	5,826,801
Local with 358 terms from each tag		**4,753,754**
Local with 50 terms from each tag		961,202
Enriched–local with 190 terms from each tag		**4,753,285**
Enriched–local with 91 terms from each tag	**27,993,308**	

6.1 Alternative Vocabulary Storage Choices

To address the shortcomings of the global vocabulary storage approach, and following the options available for feature selection, two alternative vocabulary storage approaches are examined: local and enriched–local vocabulary storage.

6.1.1 Local

The local vocabulary storage approach stores a potentially different set of terms for each materialized centroid and the number of terms stored per centroid may also vary. For a materialized centroid C_T corresponding to the set of documents S_T identified by a conjunction of tags in set T, the local vocabulary V_T^L corresponds to the union of individual tag vocabularies for the tags found in T as defined in Equation 6.1

$$V_T^L = \bigcup_{t \in T} V_t \tag{6.1}$$

where V_t is a vocabulary of an individual tag that stores the top k terms based on mutual information.

For example, a centroid $C_{databases}$ for a document set $S_{databases}$ corresponding to a single tag query "databases" has a local vocabulary $V_{databases}^L$ that uses the top k terms for

Table 6.2: Ratio of vocabulary size of tags with shared documents vs. randomly chosen tags in a) NYT (top 500 terms per tag) and b) ACM (top 100 terms per tag).

NYT

	Overlapping			**Random**			**Overlapping vs. random**
Conjunction length	Instance count	Mean ratio	Standard dev.	Instance count	Mean ratio	Standard dev.	
1	1,015	1.0	0.00	1,015	1.0	0.00	1.00
2	16,448	1.7	0.14	100,000	1.9	0.07	0.90
3	20,905	2.1	0.21	100,000	2.7	0.13	0.78
4	12,217	2.4	0.26	100,000	3.4	0.18	0.70
5	5,289	2.7	0.31	100,000	4.1	0.22	0.65
6	2,401	3.1	0.31	100,000	4.8	0.27	0.65
7	1,152	3.5	0.24	100,000	5.4	0.30	0.64
8	493	3.8	0.15	100,000	6.0	0.34	0.63
9	151	4.0	0.10	100,000	6.6	0.37	0.61
10	27	4.2	0.08	100,000	7.1	0.40	0.59
11	2	4.4	0.01	100,000	7.7	0.43	0.57

ACM

	Overlapping			**Random**			**Overlapping vs. random**
Conjunction length	Instance count	Mean ratio	Standard dev.	Instance count	Mean ratio	Standard dev.	
1	9,098	1.0	0.00	9,098	1.0	0.00	1.00
2	14,262	1.8	0.12	100,000	2.0	0.02	0.91
3	5,280	2.3	0.26	100,000	2.9	0.04	0.78
4	3,860	2.9	0.39	100,000	3.9	0.06	0.75
5	3,700	3.6	0.44	100,000	4.8	0.07	0.75
6	3,199	4.3	0.50	100,000	5.7	0.09	0.75
7	2,390	5.0	0.52	100,000	6.6	0.11	0.76
8	1,520	5.6	0.50	100,000	7.5	0.13	0.75
9	776	6.2	0.45	100,000	8.4	0.14	0.74
10	297	6.8	0.38	100,000	9.2	0.16	0.74
11	79	7.3	0.29	100,000	10.0	0.18	0.73
12	13	7.8	0.17	100,000	10.9	0.20	0.72

that tag, while a centroid $C_{databases \wedge cloud\_computing}$ for a query "databases $\wedge$ cloud computing" which is a conjunction of two tags uses local vocabulary $V^L_{databases \wedge cloud\_computing}$ that is composed of the union of the top k terms coming from vocabularies $V_{databases}$ and $V_{cloud\_computing}$.

Even though the local vocabulary for a conjunction of tags has to store vocabularies for multiple tags, the increase in vocabulary size is not substantial if the tags involved have at least several shared documents. This is in contrast to the size of vocabulary that would result when combining vocabularies for randomly chosen set of tags. Table 6.2 shows how the vocabulary size of a centroid varies with the number of tags in the conjunction of tags that have more than the threshold number of shared documents (referred to as "Overlapping") and contrasts it to the size of vocabularies for randomly chosen sets of tags (referred to as "Random"). The "Conjunction length" column refers to the number of different tags that are being considered; "Instance count" corresponds to the number of tag sets that are considered, for "Overlapping" we consider the same tag sets that were previously identified in Table 3.1, while for "Random" we consider 100,000 random combinations of tag sets; the "Mean ratio" column measures the ratio of the size of union of the vocabularies of chosen tag sets to the size of individual tag vocabulary k and is expressed as $\frac{\left|\bigcup_{t \in T} V_t\right|}{k}$.

The smaller the "Mean ratio", the larger the overlap between the vocabularies of the tags considered. The "Overlapping vs. random" column displays the ratio of these means. The vocabularies for a set of tags with overlapping documents have a noticeable amount of shared vocabulary in contrast to when a random set of tags are considered.

In addition, although an increase in the number of tags in the conjunctive queries results in a larger vocabulary that is available for representing the centroid, in reality only a small subset of that vocabulary has non-zero values. This is because as more tags are used in a conjunction, there are fewer remaining documents and their focus is narrowed down and more specialized. When documents share a large number of tags, this usually implies that the documents are very similar to each other and in turn are using very similar vocabularies. As a result, a large portion of the available vocabulary does not appear in any of the remaining documents and in turn the centroid must store fewer terms with non-zero values. The centroid storage approach proposed in Section 6.2 allows for an efficient storage of such centroids.

6.1.2 Enriched–local

The local vocabulary storage approach is designed to provide a rich vocabulary for answering single tag or conjunctive queries, which are expected to be dominant. However, since users can pose disjunctive queries or have them automatically generated by a system through query expansion, the vocabulary available for such queries should also be rich enough to provide meaningful indicative terms. In order to address this concern we develop an enriched–local vocabulary storage approach.

The enriched–local vocabulary storage approach is a modification to the local vocabulary storage. For each materialized centroid, in addition to terms stored by the local vocabulary approach, it stores all the terms that come from the tags that are known to co-occur with the tags found in the tag conjunction defining the centroid. The set of tags from which the terms are included for the centroid is derived using Equation 6.3.

$$N(T) = \bigcap_{t \in T} getMaterializedConjuncts(t) \tag{6.2}$$

Table 6.3: Co-occurring tags that result in materialized centroids for conjunctions

Tag	**Co-occurring tag list**
t_1	$t_1, t_2, t_3, t_4, t_5, t_6$
t_2	$t_1, t_2, t_3, t_4, t_7, t_8$
t_3	$t_1, t_2, t_3, t_4, t_5, t_7$

Table 6.4: Enriched–local storage vocabularies for various materialized centroids along with the sets of tag vocabularies used to form them.

Enriched–local vocabulary	**Set of tag vocabularies used**
$V^E_{t_1}$	$V_{t_1} \cup V_{t_2} \cup V_{t_3} \cup V_{t_4} \cup V_{t_5} \cup V_{t_6}$
$V^E_{t_2}$	$V_{t_1} \cup V_{t_2} \cup V_{t_3} \cup V_{t_4} \cup V_{t_7} \cup V_{t_8}$
$V^E_{t_3}$	$V_{t_1} \cup V_{t_2} \cup V_{t_3} \cup V_{t_4} \cup V_{t_5} \cup V_{t_7}$
$V^E_{t_1 \wedge t_2}$	$V_{t_1} \cup V_{t_2} \cup V_{t_3} \cup V_{t_4}$
$V^E_{t_1 \wedge t_3}$	$V_{t_1} \cup V_{t_2} \cup V_{t_3} \cup V_{t_4} \cup V_{t_5}$
$V^E_{t_2 \wedge t_3}$	$V_{t_1} \cup V_{t_2} \cup V_{t_3} \cup V_{t_4} \cup V_{t_7}$
$V^E_{t_1 \wedge t_2 \wedge t_3}$	$V_{t_1} \cup V_{t_2} \cup V_{t_3} \cup V_{t_4}$

$$V^E_T = \bigcup_{t \in N(T)} V_t \tag{6.3}$$

where T corresponds to a set of tags used in a conjunction that identifies a materialized centroid, *getMaterializedConjuncts*() returns a set of tags that together with tag t have a materialized conjunction, V_t is vocabulary for tag t.

Given three tags: t_1, t_2, t_3; with tag co-occurrence relationships specified in Table 6.3, the enriched–local storage materialization strategy would assign to each of the centroids a vocabulary that is the union of vocabularies for the listed set of tags as shown in Table 6.4.

The use of co-occurring tags and their vocabularies to enrich the vocabularies of materialized centroids can be problematic when there exist tags that co-occur with many others. This phenomenon is observed in both the NYT and ACM collection as is shown in Figure 3.4, where there exist a few tags that co-occur with many other tags. However, the majority of tags co-occur with only a few tags, and those that co-occur with many tags have a limited impact on the vocabulary size due to having many common terms (Table 6.2). As a result, a materialization strategy that uses enriched–local vocabulary storage is feasible for storing the NYT and ACM collections, as is shown in Table 6.1. However, the strategy requires that the local vocabulary that is contributed by each tag be reduced from 358 terms to 190, for the ACM; and from 796 to 91, for the NYT. This reduction is necessary to have the enriched–local vocabulary storage consumption aligned with the other approaches.

6.2 Efficient Storage of Set Centroid

For a set of tags T defining a conjunction, the centroid C_T is defined in terms of vocabulary V_T. However, when the size of vocabulary V_T is large, it is possible that many of the terms in C_T have a value of zero. Such a situation can occur when using global, local or enriched–local vocabulary. To avoid wasting unnecessary space storing vocabulary terms with zero values, we define centroid P_T that only stores values for terms present in the centroid. The terms absent from P_T but found in the vocabulary V_T have an implicit value of zero. Given V_T and centroid P_T, the centroid C_T is derived using Equations 6.4 and 6.5.

$$C_T[term].sum = \begin{cases} 0 & \text{if } term \in V_T \land term \notin P_T \\ P_T[term].sum & \text{if } term \in P_T \end{cases} \tag{6.4}$$

$$C_T[term].count = \begin{cases} 0 & \text{if } term \in V_T \land term \notin P_T \\ P_T[term].count & \text{if } term \in P_T \end{cases} \tag{6.5}$$

For the local and enriched–local vocabulary storage techniques, all the vocabularies for the materialized centroids can be stored as bitmap entries in a database table and compressed using World-Aligned Hybrid (WAH) encoding [Lemire et al., 2010] to minimize the consumed space. The zero values for terms in a centroid C_T are retrieved by comparing the terms found in the centroid P_T to those stored in the vocabulary table. The P_T data is stored as tables.

6.3 Available Vocabulary for Query Result

As described in Chapter 5, the individual cell materialization strategy I(T) stores a set of centroids that satisfy Equation 5.1. As a result, when relying on I(T), in order to calculate a centroid C_Q for a query Q, C_Q needs to be transformed into an expanded form through the use of the inclusion–exclusion principle, so that each term in the expanded form satisfies the '*alltags*' constraint. Once the query is in this expanded form, it can be answered by aggregating the centroid values for all corresponding terms.

Table 6.5: Transformation steps in a query

Query	**Transformation**
$C_{t_1 \wedge (t_2 \vee \bar{t_3})}$	Original
$C_{(t_1 \wedge t_2) \vee (t_1 \wedge \bar{t_3})}$	Disjunctive normal form
$C_{t_1 \wedge t_2} + C_{t_1 \wedge \bar{t_3}} - C_{t_1 \wedge t_2 \wedge \bar{t_3}}$	Inclusion–exclusion principle
$C_{t_1 \wedge t_2} + (C_{t_1} - C_{t_1 \wedge t_3}) - (C_{t_1 \wedge t_2} - C_{t_1 \wedge t_2 \wedge t_3})$	Negation removal
$C_{t_1} - C_{t_1 \wedge t_3} + C_{t_1 \wedge t_2 \wedge t_3}$	Query in form answerable by materialization model
$C_{t_1 \vee t_2}$	Original
$C_{t_1} + C_{t_2} - C_{t_1 \wedge t_2}$	Query in form answerable by materialization model

For example, when given a query $t_1 \wedge (t_2 \vee \bar{t_3})$, the centroid $C_{t_1 \wedge (t_2 \vee \bar{t_3})}$ is calculated by aggregating centroids C_{t_1}, $C_{t_1 \wedge t_3}$ and $C_{t_1 \wedge t_2 \wedge t_3}$; when given a query $t_1 \vee t_2$, the centroid $C_{t_1 \vee t_2}$ is calculated by aggregating centroids C_{t_1},C_{t_2},$C_{t_1 \wedge t_2}$ as shown in Table 6.5. Additional examples of queries in this expanded form can be found in Table 5.1.

In all three materialization strategies proposed in Chapter 5 (TM, TMA, MCC), only a subsets of cells are materialized. As a result, some of the centroids are obtained from materialized cells, while others are derived by aggregating individual DTVs. For example in the case of TM, only centroids with document count above the materialization threshold are materialized. Based on this property, queries issued to the system can be classified by the number of materialized centroids found in their expanded form (that need to be aggregated to produce a requested centroid). Given a query Q, the resulting centroid C_Q is derived by aggregating centroids found in X_Q, where X_Q is the set of centroid terms found in expanded form of C_Q after the inclusion–exclusion principle transformation and negation removal have been applied to C_Q. The function $M(X_Q)$ returns a set of materialized centroids found in X_Q. If $|M(X_Q)| = 0$, Q is classified as a query with zero materialized centroids; if $|M(X_Q)| = 1$, Q is classified as a query with one materialized centroid; and if $|M(X_Q)| \geq 2$, Q is classified as query with many materialized centroids. These classes are examined in more detail in the following subsections.

When local or enriched–local vocabulary storage is used, an increase in the number of materialized centroids that need to be aggregated can result in a reduction of available vocabulary for representing the derived centroid. This vocabulary reduction is attributed to the fact that only the shared vocabulary that is found in all materialized centroids can be used. Meanwhile, each materialized centroid can potentially have a distinct vocabulary, which can lead to few terms that are found in all materialized centroids. On the other hand, since the global vocabulary approach uses a common vocabulary, there are no complications associated with aggregating multiple materialized centroids.

6.3.1 Zero Materialized Centroids

Given a query Q, the resulting centroid C_Q is computed entirely from aggregating the corresponding set of DTVs found in S_Q. Since C_Q is calculated fully from the DTVs which did not have any terms filtered out, its vocabulary V_Q has all terms available for representing the centroid.

6.3.2 One Materialized Centroid

Given a query Q, the resulting centroid C_Q is calculated by aggregating the single centroid $C_x \in M(X_Q)$ with all DTVs of documents found in $X_Q - M(X_Q)$ (the non-materialized centroids) to produce C_Q. The use of the materialized centroid C_x in the aggregation reduces the aggregation cost at the expense of reducing the available vocabulary V_Q for representing C_Q, which is set to V_x, where V_x is the vocabulary available to C_x.

For centroid $C_{t_1 \wedge (t_2 \vee \bar{t_3})}$, calculated by aggregating centroids C_{t_1}, $C_{t_1 \wedge t_3}$, and $C_{t_1 \wedge t_2 \wedge t_3}$, if only C_{t_1} is materialized then the vocabulary $V_{t_1 \wedge (t_2 \vee \bar{t_3})}$ available for representing $C_{t_1 \wedge (t_2 \vee \bar{t_3})}$ is limited to the vocabulary found in V_{t_1}. Vocabulary present in DTVs but not in V_{t_1} is removed to avoid producing incorrect values for these terms in the centroid $C_{t_1 \wedge (t_2 \vee \bar{t_3})}$. Otherwise, an incorrect value would be assigned to $C_{t_1 \wedge (t_2 \vee \bar{t_3})}[w]$ if there is a document $d \in S_{t_1}$ such that $\exists w \in V_d \wedge w \notin V_{t_1} \wedge w \in (V_{t_1 \wedge t_3} \cup V_{t_1 \wedge t_2 \wedge t_3})$.

6.3.3 Many Materialized Centroids

Given a query Q, the resulting centroid C_Q is calculated by aggregating all centroids $C_x \in M(X_Q)$ with all DTVs of documents found in $X_Q - M(X_Q)$ (the non-materialized centroids) to produce C_Q. Since there exist multiple centroids in $M(X_Q)$, the vocabulary V_Q available for representing C_Q is derived by Equation 6.6.

$$V_Q = \bigcap_{q \in M(X_Q)} V_q \tag{6.6}$$

As an example, in the case of centroid $C_{t_1 \wedge (t_2 \vee \bar{t_3})}$ if all three centroids from which it is derived are materialized, then the vocabulary $V_{t_1 \wedge (t_2 \vee \bar{t_3})}$ available for representing

Table 6.6: Set of tag vocabularies available for answering query based on local and enriched–local storage approach.

Query	Aggregation	Local storage	Enriched–local storage
$t_1 \vee t_2$	$C_{t_1} + C_{t_2} - C_{t_1 \wedge t_2}$	$V_{t_1} \cap V_{t_2}$	$V_{t_1} \cup V_{t_2} \cup V_{t_3} \cup V_{t_4}$
$t_1 \vee t_3$	$C_{t_1} + C_{t_3} - C_{t_1 \wedge t_3}$	$V_{t_1} \cap V_{t_3}$	$V_{t_1} \cup V_{t_2} \cup V_{t_3} \cup V_{t_4} \cup V_{t_5}$
$t_2 \vee t_3$	$C_{t_2} + C_{t_3} - C_{t_2 \wedge t_3}$	$V_{t_2} \cap V_{t_3}$	$V_{t_1} \cup V_{t_2} \cup V_{t_3} \cup V_{t_4} \cup V_{t_7}$
$t_1 \vee t_2 \vee t_3$	$C_{t_1} + C_{t_2} + C_{t_3} - C_{t_1 \wedge t_2}$ $-C_{t_1 \wedge t_3} - C_{t_2 \wedge t_3} + C_{t_1 \wedge t_2 \wedge t_3}$	$V_{t_1} \cap V_{t_2} \cap V_{t_3}$	$V_{t_1} \cup V_{t_2} \cup V_{t_3} \cup V_{t_4}$
$t_1 \wedge \bar{t_3}$	$C_{t_1} - C_{t_1 \wedge t_3}$	V_{t_1}	$V_{t_1} \cup V_{t_2} \cup V_{t_3} \cup V_{t_4} \cup V_{t_5}$
$t_1 \wedge (t_2 \vee \bar{t_3})$	$C_{t_1} - C_{t_1 \wedge t_3} + C_{t_1 \wedge t_2 \wedge t_3}$	V_{t_1}	$V_{t_1} \cup V_{t_2} \cup V_{t_3} \cup V_{t_4}$

$C_{t_1 \wedge (t_2 \vee \bar{t_3})}$ is limited to the vocabulary found in $V_{t_1} \cap V_{t_1 \wedge t_3} \cap V_{t_1 \wedge t_2 \wedge t_3}$. Terms absent from this vocabulary are removed from centroid $C_{t_1 \wedge (t_2 \vee \bar{t_3})}$ to avoid computing incorrect values for them.

In a disjunctive query Q, the size of X_Q is equal to $2^n - 1$, where n is the number of tags in Q. As the size of X_Q grows, often the size of $M(X_Q)$ grows as well. As the size of $M(X_Q)$ grows, the size of vocabulary V_Q can be drastically reduced in size when there is little shared vocabulary among the $V_i \in M(X_Q)$, since V_Q is defined by the intersections of all vocabularies found in $M(X_Q)$. For example, a query $t_1 \vee t_2 \vee t_3$ will have $2^3 - 1$ centroids in the expanded form, which implies $0 \leq |M(X_{t_1 \vee t_2 \vee t_3})| \leq 7$ and results in a corresponding number of vocabularies intersected.

When there are multiple materialized centroids in the expanded form on which C_Q relies, there may be a significant difference in the available vocabulary for V_Q depending on whether local vocabulary V_Q^L or enriched–local vocabulary V_Q^E is used. For example, given a centroid $C_{t_1 \vee t_2}$ defined by query $t_1 \vee t_2$, when local vocabulary $V_{t_1 \vee t_2}^L$ is used, the vocabulary is limited to $V_{t_1} \cap V_{t_2} \cap V_{t_1 \wedge t_2} = V_{t_1} \cap V_{t_2} \cap (V_{t_1} \cup V_{t_2}) = V_{t_1} \cap V_{t_2}$. Alternatively, when enriched–local vocabulary $V_{t_1 \vee t_2}^E$ is used with tag co-occurrences defined in Table 6.3, the available vocabulary is limited to $V_{t_1} \cup V_{t_2} \cup V_{t_3} \cup V_{t_4}$. As another example, for query $t_1 \wedge (t_2 \vee \bar{t_3})$, the local vocabulary $V_{t_1 \wedge (t_2 \vee \bar{t_3})}^L$ is limited to V_{t_1}, while enriched–local vocabulary $V_{t_1 \wedge (t_2 \vee \bar{t_3})}^E$ is limited to $V_{t_1} \cup V_{t_2} \cup V_{t_3} \cup V_{t_4}$.

Additional examples of available vocabularies for various queries when local and enriched–local vocabularies are used are shown in Table 6.6. The enriched–local vocabulary approach provides a larger available vocabulary for representing centroids.

6.4 Incorporating Vocabulary Storage into MCC Partial Materialization

The previously proposed MCC partial materialization strategy was designed on top of a global vocabulary and did not take into consideration the use of local or enriched–local vocabulary storage approaches. Below we show how both of these vocabulary storage approaches can be used with MCC.

6.4.1 Cell Vocabulary

The MCC strategy relies on materialization of centroid C_P for each artificial cell $P \in \mathbb{P}$, where $\mathbb{P}$ is a set of derived artificial cells. For each cell P there exist a set of $\mathbb{J}_P$ tag conjunctions (corresponding to cells identified to have '*alltags*') such that for each $J \in \mathbb{J}_P$, the centroid C_J is derived from C_P by adding or removing DTVs found in individual documents. Each $J \in \mathbb{J}_P$ can involve conjunction of different tags and in turn require a different vocabulary. To ensure that all C_J can be derived from C_P, the vocabulary V_P must contain the vocabularies of all V_J as defined in Equation 6.7.

$$V_P = \bigcup_{J \in \mathbb{J}_P} V_J \tag{6.7}$$

Since the materialized centroid C_P uses vocabulary V_P, V_P can be used for all C_J such that $J \in \mathbb{J}_P$. Since $V_J \subseteq V_P$ for all $J \in \mathbb{J}_P$, centroid C_J benefits from having a larger vocabulary as a side effect of satisfying the MCC requirements.

6.4.2 Generating Materialization Plan

In Section 5.3.3, we developed the MCC partial materialization strategy, which relies on Algorithm 5 to produce a set of partitions $\mathbb{P} = \{P_1, ..., P_n\}$. For each partition $P \in \mathbb{P}$, a set of documents S_P that belongs to the partition is determined. For each set S_P, the corresponding centroid C_P needs to be calculated and materialized. In this section, we show how such materialization is performed when using a local vocabulary for representing each centroid. This is easily extended to support enriched–local vocabularies instead.

The materialization process is divided into two steps:

1. generation of materialization plan,

2. materializing centroids based on the materialization plan.

Algorithm 6 is used to generate a materialization plan $\mathbb{M}$ for all centroids C_P, where $P \in \mathbb{P}$, under the assumption that a local vocabulary is used. The algorithm takes as input $\mathbb{P}$ and the mapping $\mathbb{J}$ that specifies the set of tag conjunctions $\mathbb{J}_P$ that rely on each $P \in \mathbb{P}$. The algorithm works by retrieving each P, one by one, in ascending order based on the number of documents belonging to the partition $|S_P|$. For each partition P, the set of tag conjunctions $\mathbb{J}_P$ that are derived from it, are retrieved from $\mathbb{J}$.

The tag conjunctions $J \in \mathbb{J}_P$ are used to define a parent-child relationship between the centroids C_P for $P \in \mathbb{P}$, such that the centroid C_{P_i} can be used to help with computation of C_{P_j}, where $P_i, P_j \in \mathbb{P}$. For example, if $\mathbb{J}_{P_i}$ contains $t_1 \wedge t_2$ and $\mathbb{J}_{P_j}$ contains t_1, then centroid C_{P_i} may be used to help compute C_{P_j} since document set S_{t_1} is a superset of $S_{t_1 \wedge t_2}$. For each P, all the candidate partitions that may help with reducing the cost of aggregation of C_P are put into a list L. The mapping of materialized partition centroids and which tag conjunctions can benefit from them is stored in T, which serves as a lookup table.

For each P, the materialization plan M_P is determined by first checking if there already are some pre-materialized partitions held in L which could be leveraged to speed up the aggregation process. If L is empty then C_P is calculated by aggregating all the documents found in S_P. Otherwise, L is scanned to find suitable candidates l whose centroid C_l can be used towards the computation of C_P and in turn reduce the number of individual documents that need to be aggregated. For a centroid C_l, where $l \in L$, to be considered a suitable candidate for use as part of the materialization plan M_P, it needs to significantly reduce the number of documents that need to be fetched. It also needs to have a vocabulary that is a superset of V_P, which is ensured when the centroid in question is used for deriving centroids of tag conjunctions that are a superset of those that are answered by C_P. Multiple candidate partitions are picked from L in a greedy manner so as to provide a maximum reduction in the number of individual documents that need to be aggregated to derive C_P.

Since the materialization plan $\mathbb{M}$ for the set of partitions $\mathbb{P}$ is generated in ascending order of $|S_P|$, the centroids that are processed at the early stage of the materialization plan can be used as part of the materialization plan of bigger centroids, which are processed later in the plan.

Once the materialization plan $\mathbb{M}$ is determined, Algorithm 7 uses this plan to calculate the centroids for all C_P, where $P \in \mathbb{P}$. As the first step in materialization of each centroid C_P, function *getAggregationOfElements*() is called to aggregate all centroids C_r where $r \in M_P.block$ with the document term vectors of documents listed in $M_P.docs$ to produce

Algorithm 6 Generating materialization plan

```
Input: $\mathbb{P}$, the partitions requiring materialized centroids; $\mathbb{J}$, an indexed set of sets of tag conjunctions supported by each partition
Returns: Materialization Plan $\mathbb{M}$
$\mathbb{M} \leftarrow \emptyset$, $\forall_{J \in \mathbb{J}} T[J] \leftarrow \emptyset$
for $P \in getSortedByAscDocCount(\mathbb{P})$ do
    $L \leftarrow \emptyset$, $M_P \leftarrow \emptyset$
    for $J \in \mathbb{J}_P$ do        ▷ partitions potentially useful for calculating $C_P$ are put in $L$.
        $L \leftarrow L \cup T[J]$
    end for
    $M_P.docs \leftarrow S_P$
    if $L \neq \emptyset$ then
        $t_P \leftarrow getTags(\mathbb{J}_P)$        ▷ retrieves all the tags found in conjunctions that rely on $P$.
        for $l \in getSortedByDescDocCount(L)$ do        ▷ greedy search for useful building blocks.
            $M_{stored} \leftarrow \mathbb{M}[l]$
            $t_{stored} \leftarrow getTags(\mathbb{J}_l)$
            if $t_P.isSubSetOf(t_{stored})$ then        ▷ only candidates with richer vocabulary are considered.
                if $\frac{|M_P.docs| - |M_P.docs - M_{stored}.getDocs()|}{|M_{stored}.getDocs()|} > 0.5$ then        ▷ significant reduction required.
                    $M_P.block \leftarrow M_P.block \cup l$
                    $M_P.docs \leftarrow M_P.docs - M_{stored}.getDocs()$
                end if
            end if
        end for
    end if
    $\mathbb{M}.add(P, M_P)$
    for $J \in \mathbb{J}_P$ do        ▷ generates a list of potential building block candidates for tag conjunctions.
        $parents \leftarrow getParentCombinationsOfCell(J)$
        for $p \in parents$ do
            $T[p].enque(P)$
        end for
    end for
end for
return $\mathbb{M}$
```

a cluster term vector C_P. The vocabulary V_P is then trimmed by removing all terms whose global frequency is below 5. As an additional vocabulary trimming step, all the tag conjunctions $J \in \mathbb{J}_P$ are retrieved and used to determine the local vocabulary that V_P needs to support, and everything else is filtered out. After all the filters have been applied to the V_P, centroid C_P is saved in $\mathbb{C}$.

Algorithm 7 Instantiating the materialization plan

```
  **Input:** Materialization Plan $\mathbb{M}$
  **Returns:** List of Materialized Clusters $\mathbb{C}$
  $\mathbb{C} \leftarrow \emptyset$
  **for** $(P, M) \in \mathbb{M}$ **do**
      $C_P \leftarrow getAggregationOfElements(M)$
      $C_P.removeGlobalInfrequentTerms()$
      $tags \leftarrow getTags(\mathbb{J}_P)$
      $V_P \leftarrow getLocalDictionaryForTags(tags)$
      $C_P.applyDictionaryFilter(V_P)$
      $\mathbb{C}.enque(C_P)$
  **end for**
  **return** $\mathbb{C}$
```

6.5 Evaluation

As stated at the beginning of this chapter, the vocabulary available for answering queries affects the set of indicative terms and indicative documents that will be returned to users. We have chosen to use burstiness to select indicative terms (see Section 2.4.2). For any set of documents, in order to guarantee that the best bursty terms are picked, it would be necessary to store term occurrence statistics for all terms found in the document set. By storing the full global vocabulary V^F, each set of documents S_Q specified by query Q has vocabulary V_Q^F available to derive the bursty terms or pick indicative documents. However, since storing the whole vocabulary is not practical because of its size, we want to make sure that for any set of documents S_Q, the vocabulary V_Q available for the set contains the top k terms found in V_Q^F based on mutual information. We evaluate the three vocabulary storage approaches—global, local, and enriched–local—on their ability to provide the top k terms for various queries Q. The recall performance of a vocabulary storage approach V^A on providing the top k terms found in V_Q^F for query Q is measured using *k-precision* [Candan and Sapino, 2010] defined in Equation 6.8, and normalized Discounted Cumulative Gain ($nDCG_k$) [Candan and Sapino, 2010] defined in Equation 6.10.

$$precision(k, Q, A) = \frac{|V_Q^F[1:k] \cap V_Q^A|}{k} \tag{6.8}$$

where $V_Q^F[1:k]$ returns a set of top k terms based on mutual information for the set of documents S_Q.

$$DCG_k = rel_1 + \sum_{i=2}^{k} \frac{rel_i}{\log_2 i} \tag{6.9}$$

$$nDCG_k = \frac{DCG_k}{IDCG_k} \tag{6.10}$$

where rel_i is the relevance score of term i, and corresponds to the mutual information of the term. $IDCG_k$ is the Discounted Cumulative Gain for V_Q^F. For our analysis, we assume that k is set to 100 for the NYT and 50 for the ACM. A smaller k is chosen for the ACM collection since it has a much smaller vocabulary than the NYT (13,015 terms for the ACM vs. 238,065 terms for the NYT), has a larger number of tags (9,098 for the ACM vs. 1,015 for the NYT), has less documents (66 thousand for the ACM vs 1.5 million for the NYT), and many of the tag defined document sets have few documents. All these factors contribute to the existence of fewer set specific terms in the ACM on which we can evaluate the effectiveness of the different storage approaches.

In order to perform a fair comparison of the three vocabulary storage approaches, the vocabulary size of each approach is configured such that the total storage space consumed by a MCC partial materialization with each vocabulary storage approach is roughly the same. Table 6.1 shows the total storage cost of using MCC partial materialization with each of the three vocabulary storage techniques under various vocabulary sizes.

The global vocabulary storage approach is configured to use 2,479 terms for the NYT collection, which are generated by taking the top 5 terms based on mutual information for each of the 1,015 tags, and 3,635 terms for the ACM collection which are generated by taking the top single term from each of the 9,098 tags based on mutual information. The local vocabulary storage approach is configured to store the top 796 terms per tag for the NYT and 358 for the ACM collection. The enriched–local vocabulary storage approach is configured to store the top 91 terms per tag for the NYT and 190 terms per tag for the ACM collection. (A smaller number of terms per tag are chosen with the enriched–local approach than the local approach because in the former approach more tags contribute to the vocabulary of each materialized centroid.)

6.5.1 Effect of Vocabulary Storage Approach on Representative Terms

In Table 6.7, we illustrate how the choice of vocabulary storage approach affects the set of indicative terms shown to the user as a summary for a document set satisfying the query "*Finances AND Personal Finances AND Bankruptcies.*" The indicative terms available for summarizing the set are compared against the gold standard in which the full vocabulary is stored. Only the local storage vocabulary approach is able to provide the same set of indicative terms in its summary as the gold standard. For the "Other terms" found by the enriched-local and global vocabulary storage approaches, we indicate the ranking of the term as found in the full vocabulary.

6.5.2 Evaluating Term Coverage

The three vocabulary storage approaches are evaluated on over 400,000 queries which vary in length and in the types of operators used. Four types of queries are examined:

1. Single tag queries.
2. Conjunction queries ranging in length from 2-12.
3. Disjunction queries ranging in length from 2-12.
4. Queries containing a mix of conjunctions and disjunctions, and varying in length from x to y.

Single Tag Queries

Analysis of the three vocabulary storage approaches on all possible single tag queries for both the NYT and ACM collection are shown in Table 6.8. The analysis confirms that the global vocabulary storage approach provides inferior performance.

Conjunction Queries

In the analysis of all conjunctive queries with significant overlap of documents (at least 50 for the NYT, and 5 for the ACM), the global vocabulary approach had the lowest recall, while the local approach had the highest recall and was near optimal. The details of the comparison are summarized in Table 6.9.

Table 6.7: Comparing the top 20 indicative terms returned for query "*Finances AND Personal Finances AND Bankruptcies*" when relying on full, local, enriched–local, or global vocabularies.

Rank	**Storage**			
	Full	Local	Enriched–local	Global
1	chewco*	✓		
2	receivership*	✓		
3	causei*	✓		
4	debtor*	✓	✓	
5	filer*	✓		
6	campeau*	✓		
7	ljm*	✓		
8	bankruptci*	✓	✓	✓
9	enron*	✓	✓	✓
10	fastow*	✓	✓	
11	creditor*	✓	✓	✓
12	sherron*	✓		
13	repai*	✓	✓	
14	solvent*	✓		
15	andersen*	✓	✓	
16	kopper*	✓		
17	repay*	✓	✓	
18	auditor*	✓	✓	✓
19	petit*	✓		
20	chapter*	✓	✓	✓
Match count	20	20	10	5
Other terms			file*(22),debt*(23), ow*(24),auction*(27), unsecur*(28),sec*(29), reorgan*(30), lender*(32), bankrupt*(34), liabil*(37)	file*(22),debt*(23), auction*(27),sec*(29), reorgan*(30),lender*(32), partnership*(36), liabil*(37),asset*(40), charit*(48),treasuri*(50), mortgag*(53),code*(55), estat*(56),wife*(60)

Table 6.8: Coverage analysis on single tag queries for the three vocabulary storage approaches.

Collection	**k**	**Query count**	*k-precision*			**nDCG$_k$**		
			Local	**Global**	**Enriched local**	**Local**	**Global**	**Enriched local**
NYT	100	1,015	100	49.3	96.5	1.000	0.868	0.999
ACM	50	9,098	100	47.4	100	1.000	0.846	1.000

Table 6.9: Coverage analysis on conjunctive queries for the three vocabulary storage approaches on the NYT (top 100) and the ACM (top 50)

Collection	**Tag count**	**Query count**	*k-precision*			**nDCG$_k$**		
			Local	**Global**	**Enriched local**	**Local**	**Global**	**Enriched local**
NYT	2	16,448	87.5	46.6	75.6	0.951	0.748	0.903
	3	20,905	89.6	47.8	77.8	0.960	0.748	0.914
	4	12,217	93.1	49.4	81.1	0.976	0.753	0.932
	5	5,289	96.5	50.7	84.1	0.989	0.746	0.949
	6	2,401	98.5	50.2	85.1	0.995	0.715	0.957
	7	1,152	99.5	48.8	84.9	0.999	0.680	0.958
	8	493	99.9	48.1	84.9	1.000	0.662	0.959
	9	151	100	48.1	85.2	1.000	0.659	0.959
	10	27	100	48.2	85.6	1.000	0.660	0.959
	11	2	100	48.5	86.5	1.000	0.661	0.960
ACM	2	14,262	80.8	46.4	74.0	0.950	0.790	0.931
	3	5,280	83.8	44.8	81.4	0.955	0.764	0.955
	4	3,860	94.2	41.0	92.6	0.986	0.726	0.988
	5	3,700	98.0	40.6	95.2	0.997	0.715	0.996
	6	3,199	99.0	42.4	94.4	0.999	0.721	0.996
	7	2,390	99.6	44.4	93.2	1.000	0.730	0.996
	8	1,520	99.8	45.8	92.0	1.000	0.736	0.995
	9	776	100	46.2	91.2	1.000	0.738	0.995
	10	297	100	46.2	90.6	1.000	0.737	0.995
	11	79	100	46.0	90.2	1.000	0.737	0.994
	12	13	100	46.0	90.0	1.000	0.736	0.994

Disjunctive Queries

Disjunctive queries correspond to queries in which the only operator present is "OR". The queries analyzed range in length from two to eight tags. When a long disjunctive query is transformed using the inclusion–exclusion principle it results in many centroids that need to be aggregated. By testing the vocabulary storage approaches on such queries, we can learn how well they handle scenarios where the query spans many vocabularies.

For this test we only performed disjunctions on tags that are known to have significant document overlaps and correspond to the sets of tags whose conjunctions were materialized. Tags with no document overlap are not considered as query candidates. This decision was made because the document collections cover many distinct topics most of which share very little vocabulary among each other (Table 6.2), performing a union on such sets is not expected to provide any meaningful summary or insight. For example, when a person is examining articles in the NYT, we would not expect them to request a summary on a union of articles about "Stocks and Bonds" and "Food Recipes".

As shown in Table 6.10, the local vocabulary storage approach performs poorly while the enriched–local is performing well, even for long disjunctions. The main reason for this probably stems from the local vocabulary storage approach having a very small vocabulary available for these queries, as is shown in Table 6.11. The "Cell count" column indicates the number of centroids $|X_Q|$ that need to be aggregated.

Since most queries are short and contain few disjunctions, the poor performance of the local vocabulary approach on very long disjunctions may not be significant. However, the performance was poor even for a two tag query.

Mixed Queries

Mixed queries consist of queries that are a mix of "AND" and "OR" operators. Such queries are of interest because they can be a product of an automatic query expansion performed by a system such as PubMed [Lu et al., 2009]. Each set of queries that was used for analysis involved tags that had a significant document overlap.

Three types of mixed queries are considered:

1. Disjunction of two tags joined with the rest of tags, e.g., $(t_1 \vee t_2) \wedge t_3 \wedge t_4$
2. Disjunction of three tags joined with the rest of tags, e.g., $(t_1 \vee t_2 \vee t_3) \wedge t_4$
3. Disjunction of four tags joined with the rest of tags, e.g., $(t_1 \vee t_2 \vee t_3 \vee t_4) \wedge t_5$

Table 6.10: Coverage analysis on a sample of disjunctive queries for the three vocabulary storage approaches on the NYT (top 100) and the ACM (top 50)

Collection	**Tag count**	**Query count**	*k-precision*			**nDCG**$_k$		
			Local	**Global**	**Enriched local**	**Local**	**Global**	**Enriched local**
NYT	2	1,182	44.4	57.3	97.5	0.580	0.836	0.995
	3	2,135	45.7	62.4	98.8	0.606	0.853	0.997
	4	1,555	44.3	65.0	98.9	0.599	0.859	0.997
	5	855	37.8	66.6	98.6	0.532	0.862	0.996
	6	518	27.3	67.0	98.0	0.400	0.863	0.994
	7	307	19.3	66.8	97.3	0.289	0.862	0.992
	8	142	15.2	66.8	96.4	0.228	0.861	0.989
ACM	2	1,766	56.2	75.2	100	0.797	0.951	1.000
	3	1,689	50.2	77.2	100	0.795	0.960	1.000
	4	1,605	34.2	76.4	100	0.664	0.967	1.000
	5	1,615	23.6	77.8	100	0.574	0.973	1.000
	6	1,072	17.8	78.8	100	0.526	0.976	1.000
	7	273	17.2	81.0	100	0.509	0.980	1.000
	8	58	14.6	81.2	100	0.480	0.982	1.000

Table 6.11: Size of available vocabulary for a sample of disjunctive queries.

Collection	**Tag count**	**Query count**	**Enriched local**	**Local**	**Cell count** ($\lvert X_Q \rvert$)
NYT	2	1,182	1,121.7	230.0	3
	3	2,135	1,419.4	188.8	7
	4	1,555	1,368.3	163.0	15
	5	855	1,168.3	127.2	31
	6	518	1,040.4	87.2	63
	7	307	961.0	61.9	127
	8	142	855.5	50.9	255
ACM	2	1,766	700.7	119.3	3
	3	1,689	1,112.5	63.8	7
	4	1,605	1,215.0	40.4	15
	5	1,615	1,214.1	24.1	31
	6	1,072	1,215.8	15.7	63
	7	273	1,284.9	13.5	127
	8	58	1,296.1	9.7	255

As shown in Table 6.12, the local vocabulary approach performs best on such mixed queries and is thus able to handle limited query expansions on a single tag. The enriched–local approach performed almost as well, and for queries with few conjunctions, it usually slightly outperformed the local approach. When there are many conjunctions, the result includes a small set of documents which are more uniform, and as a result, can be effectively handled by the local vocabulary storage approach. Both the local and enriched–local approaches maintained a large vocabulary from which the optimal terms could be retrieved, as is shown in Table 6.13.

We also tested the three approaches on a fourth form of mixed query:

4. Conjunctions of disjunctive tag pairs, e.g., $(t_1 \vee t_2) \wedge (t_3 \vee t_4)$

On this type of mixed query, the enriched–local approach outperformed the local approach in most situations, especially when an even number of tags were used disjunctively. The results for this are shown in Tables 6.14 and 6.15.

The recall analysis on all these queries clearly indicates that the global vocabulary storage approach performs poorly with respect to how much space it consumes. The higher recall achieved by both, the local and enriched–local vocabulary storage approaches, justifies the extra effort and infrastructure that is required to support them.

Table 6.12: Coverage analysis on a sample of mixed queries for the three vocabulary storage approaches on a) NYT (top 100) and b) ACM (top 50).

NYT

Disjunction length	**Tag count**	**Query count**	*k-precision* Global	*k-precision* Local	*k-precision* Enriched local	**nDCG$_k$** Global	**nDCG$_k$** Local	**nDCG$_k$** Enriched local
2	3	4,365	56.0	95.2	92.3	0.820	0.981	0.976
2	4	6,480	53.7	96.7	88.6	0.795	0.989	0.963
2	5	6,330	51.1	97.4	86.5	0.758	0.991	0.958
2	6	5,970	48.8	98.9	85.5	0.711	0.996	0.959
2	7	4,977	47.7	99.6	84.5	0.679	0.999	0.959
2	8	3,444	47.6	99.9	84.3	0.667	1.000	0.958
3	4	4,659	58.7	97.3	96.6	0.838	0.989	0.991
3	5	6,790	55.8	97.4	92.4	0.802	0.988	0.978
3	6	8,520	52.0	97.3	88.6	0.740	0.988	0.970
3	7	8,645	49.4	98.4	85.8	0.690	0.995	0.964
3	8	7,168	48.5	99.1	84.8	0.667	0.997	0.961
4	5	6,221	60.7	97.4	97.6	0.846	0.989	0.994
4	6	11,385	57.4	96.4	93.5	0.792	0.982	0.983
4	7	16,170	53.0	96.2	88.6	0.714	0.985	0.972
4	8	16,030	50.3	97.6	86.0	0.671	0.993	0.965

ACM

Disjunction length	**Tag count**	**Query count**	*k-precision* Global	*k-precision* Local	*k-precision* Enriched local	**nDCG$_k$** Global	**nDCG$_k$** Local	**nDCG$_k$** Enriched local
2	3	6,261	48.3	92.2	93.2	0.819	0.987	0.992
2	4	12,156	40.0	95.8	95.0	0.726	0.991	0.994
2	5	23,840	39.7	98.4	95.8	0.710	0.998	0.997
2	6	35,175	41.6	98.8	94.6	0.716	0.999	0.996
2	7	13,964	41.2	99.6	98.4	0.707	1.000	0.999
2	8	6,160	45.0	99.8	99.8	0.728	1.000	1.000
3	4	8,104	44.1	94.4	97.4	0.790	0.991	0.998
3	5	23,840	39.2	97.0	96.2	0.728	0.997	0.997
3	6	46,900	40.9	97.6	94.6	0.726	0.998	0.996
3	7	34,324	42.0	98.6	95.6	0.723	0.999	0.997
3	8	12,320	44.6	100	99.8	0.729	1.000	1.000
4	5	8,125	43.7	93.6	98.4	0.795	0.990	0.999
4	6	16,095	37.4	98.0	98.8	0.716	0.996	0.999
4	7	15,312	36.6	99.4	99.4	0.694	0.999	0.999
4	8	15,400	43.6	99.8	99.8	0.730	1.000	1.000

Table 6.13: Vocabulary size analysis on a sample of mixed queries for the three vocabulary storage approaches on a) NYT and b) ACM.

NYT

Disjunction length	**Tag count**	**Query count**	**Global**	**Local**	**Enriched local**	**Cell count** ($\|X_Q\|$)
2	3	4,365	2,479	1,141.2	1,429.7	3
2	4	6,480	2,479	1,721.5	1,572.7	3
2	5	6,330	2,479	2,358.0	1,911.5	3
2	6	5,970	2,479	2,926.6	2,539.9	3
2	7	4,977	2,479	3,250.4	3,053.8	3
2	8	3,444	2,479	3,372.0	3,464.4	3
3	4	4,659	2,479	1,044.5	1,406.3	7
3	5	6,790	2,479	1,787.2	1,539.9	7
3	6	8,520	2,479	2,553.5	2,135.6	7
3	7	8,645	2,479	3,067.9	2,790.9	7
3	8	7,168	2,479	3,279.3	3,329.1	7
4	5	6,221	2,479	1,024.9	1,278.8	15
4	6	11,385	2,479	1,844.9	1,603.8	15
4	7	16,170	2,479	2,651.5	2,473.6	15
4	8	16,030	2,479	3,100.5	3,233.6	15

ACM

Disjunction length	**Tag count**	**Query count**	**Global**	**Local**	**Enriched local**	**Cell count** ($\|X_Q\|$)
2	3	6,261	3,635	866.7	1,254.9	3
2	4	12,156	3,635	1,680.3	1,752.3	3
2	5	23,840	3,635	2,079.4	1,989.8	3
2	6	35,175	3,635	2,342.3	2,157.7	3
2	7	13,964	3,635	2,095.6	2,014.4	3
2	8	6,160	3,635	2,045.2	1,944.7	3
3	4	8,104	3,635	968.2	1,506.4	7
3	5	23,840	3,635	1,771.2	1,842.6	7
3	6	46,900	3,635	2,195.6	2,076.2	7
3	7	34,324	3,635	2,239.5	2,095.6	7
3	8	12,320	3,635	2,039.9	1,942.4	7
4	5	8,125	3,635	810.2	1,459.0	15
4	6	16,095	3,635	1,475.8	1,709.5	15
4	7	15,312	3,635	1,813.7	1,890.9	15
4	8	15,400	3,635	2,007.3	1,921.3	15

Table 6.14: Coverage analysis on a sample of mixed queries for the three vocabulary storage approaches on a) NYT (top 100) and b) ACM (top 50).

NYT

Tag count	**Query type**	**Query count**	*k-precision*			**nDCG$_k$**		
			Global	**Local**	**Enriched local**	**Global**	**Local**	**Enriched local**
4	$(t_1 \vee t_2) \wedge (t_3 \vee t_4)$	1,804	60.3	86.5	97.8	0.842	0.930	0.994
5	$(t_1 \vee t_2) \wedge (t_3 \vee t_4) \wedge t_5$	5,830	57.9	96.3	94.3	0.815	0.981	0.984
6	$(t_1 \vee t_2) \wedge (t_3 \vee t_4) \wedge (t_5 \vee t_6)$	6,685	61.5	84.3	96.5	0.821	0.901	0.990
7	$(t_1 \vee t_2) \wedge (t_3 \vee t_4) \wedge (t_5 \vee t_6) \wedge t_7$	29,420	56.1	92.7	91.2	0.739	0.962	0.979

ACM

Tag count	**Query type**	**Query count**	*k-precision*			**nDCG$_k$**		
			Global	**Local**	**Enriched local**	**Global**	**Local**	**Enriched local**
4	$(t_1 \vee t_2) \wedge (t_3 \vee t_4)$	6,684	47.4	88.1	98.4	0.812	0.979	0.999
5	$(t_1 \vee t_2) \wedge (t_3 \vee t_4) \wedge t_5$	35,904	38.1	97.5	98.7	0.723	0.995	0.999
6	$(t_1 \vee t_2) \wedge (t_3 \vee t_4) \wedge (t_5 \vee t_6)$	37,586	38.3	94.9	98.9	0.745	0.990	0.999
7	$(t_1 \vee t_2) \wedge (t_3 \vee t_4) \wedge (t_5 \vee t_6) \wedge t_7$	105,461	36.9	99.1	99.3	0.711	0.999	0.999

Table 6.15: Vocabulary size analysis on a sample of mixed queries for the three vocabulary storage approaches on a) NYT and b) ACM.

NYT

Tag count	**Query type**	**Query count**	**Cell count** ($\|X_Q\|$)	**Global**	**Local**	**Enriched local**
4	$(t_1 \vee t_2) \wedge (t_3 \vee t_4)$	1,804	9	2,479.0	705.7	1,430.7
5	$(t_1 \vee t_2) \wedge (t_3 \vee t_4) \wedge t_5$	5,830	9	2,479.0	1,530.9	1,468.6
6	$(t_1 \vee t_2) \wedge (t_3 \vee t_4) \wedge (t_5 \vee t_6)$	6,685	27	2,479.0	1,149.4	1,271.1
7	$(t_1 \vee t_2) \wedge (t_3 \vee t_4) \wedge (t_5 \vee t_6) \wedge t_7$	29,420	27	2,479.0	2,190.3	1,834.8

ACM

Tag count	**Query type**	**Query count**	**Cell count** ($\|X_Q\|$)	**Global**	**Local**	**Enriched local**
4	$(t_1 \vee t_2) \wedge (t_3 \vee t_4)$	6,684	9	3,635.0	695.4	1,372.5
5	$(t_1 \vee t_2) \wedge (t_3 \vee t_4) \wedge t_5$	35,904	9	3,635.0	1,405.2	1,642.9
6	$(t_1 \vee t_2) \wedge (t_3 \vee t_4) \wedge (t_5 \vee t_6)$	37,586	27	3,635.0	1,138.6	1,545.2
7	$(t_1 \vee t_2) \wedge (t_3 \vee t_4) \wedge (t_5 \vee t_6) \wedge t_7$	105,461	27	3,635.0	1,688.3	1,802.7

6.6 Conclusions

The NYT collection is shown to have a large vocabulary, which requires a prohibitive amount of storage space, even when only a subset of views are materialized, as is the case with the proposed MCC partial materialization. To reduce the consumed storage to a feasible size, feature selection based on the mutual information measure can be applied to reduce the size of vocabulary.

Both the NYT and ACM collections cover many topics and exhibit local vocabularies for each of those topics. This was shown to be problematic for the commonly used global vocabulary storage approach, which produced a poor vocabulary coverage for all result sets. The local and enriched–local vocabulary storage approaches were developed as alternatives to the global vocabulary storage approach to address the vocabulary locality issue and provide significant improvement in vocabulary coverage of result sets over the global approach while consuming the same amount of space. Depending on the expected query workload, either local or enriched–local approach is appropriate. Using local or enriched–local vocabulary storage increases complexity in the architecture of the system, and requires the development of rules for cell aggregation and a plan for cell materialization. However, the additional complexity in the system is justified by the amount of improvement in vocabulary coverage that is achieved, especially considering that rich vocabulary is required in order to support high quality summarization measures.

In our analysis, we limited ourselves to using mutual information. Different feature selection measures would lead to picking a different set of features. However, different features would still be expected for distinct topics, and we would expect the described vocabulary storage approaches to perform similarly when other feature selection measures are used instead.

Only a single type of global vocabulary selection approach was tested (one that picks the top k features from each tag). However, the weakness of the global vocabulary storage approach stems from its small global vocabulary being unsuitable to handle collections that cover many topics with distinct local vocabularies. As a result, even if different global vocabulary selection technique were used, it still would not be able to provide a better vocabulary coverage than the local or enriched–local approaches.

During the comparative evaluation of the three vocabulary storage approaches, we aimed to provide the best coverage for a fixed storage space. From these experiments we learned that the local and enriched–local approaches outperform the global approach. However, we did not attempt to further optimize the two approaches. For example, we did not examine how much the vocabulary size of the two approaches can be reduced while still providing an acceptable coverage. As Tables 6.11, 6.13, and 6.15 illustrate, the available vocabulary for many result sets is quite large, and it would be worthwhile to investigate if it could be reduced without having a significant impact on the resulting vocabulary coverage.

It may also be of interest to explore additional variants of the local and enriched–local approaches, such as having the tags clustered, and instead of having a separate vocabulary for each tag, have a shared vocabulary for all tags belonging to the same cluster.

Chapter 7

Enhancing Faceted Browsers

Using the infrastructure developed in previous chapters, we have developed an enhanced faceted browser interface that, in addition to letting a user perform traditional faceted browsing and searching, provides a user with summary information about the resulting document set. In our implementation, the summary includes a set of diverse documents indicative of the contained topics and a well-chosen set of indicative terms. In addition, the *information scent* [Goodwin et al., 2012] is enhanced by providing a set of indicative terms for each of the important *sub-collections* of the result set. With these summary measures in place, a user is able to quickly grasp the topics covered by large result sets.

The summary measures presented to the user are derived with the help of a partial materialization module that consumes a minimal amount of storage while providing fast access to rich and accurate summaries. The partial materialization module implements the MCC partial materialization strategy (described in Section 5.3.3) and uses the local vocabulary storage approach (described in Section 6.1.1). The utility and efficiency of the system is demonstrated on the New York Times Annotated Corpus.

7.1 Information Scent

Since faceted browsers help users navigate and explore the result sets of documents in a collection through a continuous refinement of queries (by selecting tags as filters), it is important for users to make effective use of the refinement options. It was shown by Pirolli et al. that by providing users with *information scent*, which was defined as "proximal cues to the value of distal information" [Pirolli et al., 2000], the users can navigate through the collection faster. This idea is also central to multi-menus, as defined much earlier by Raymond [Raymond, 1984]. Tunkelang suggests that "faceted search systems can increase the *information scent* associated with a refinement option by offering previews of the content associated with that selection"[Tunkelang, 2009]. For our system, we choose to provide previews by showing indicative terms.

For example, to aid a user with exploration of a set of documents tagged with "*Music*", the system provides the general summary for the corresponding set, as well as summaries for 10 additional subsets defined by intersecting a set of documents tagged with "*Music*" with another frequent tag such as "*Recordings (Audio)*" or "*Opera*". The complete summary for documents with a tag "*Music*" along with its most significant subsets is shown in Table 7.1. Summaries of additional subsets are presented in Tables 7.2, 7.3, and 7.4.

7.1.1 Sets of Interest

In order to incorporate the *information scent* into a faceted browser, we need to provide a preview for each navigational direction a user may take. Since our expectation is that a user starts with an underspecified query and then slices and dices to obtain a more specific topic by providing additional tags in a conjunction, we need to provide a summary preview for each of the possible subsets the user may want to explore next.

A set of documents S_Q defined by query Q, contains a set of tags T with which the documents of S_Q are tagged. Each tag $t \in T$ is a potential candidate to request a new set $S_{Q \wedge t}$. As a result, there are $|T|$ potential sets into which a user may want to navigate, for which we need to provide a preview. Since $|T|$ can be large, it may be both expensive for the system to generate and overwhelming for the user to consume previews for all possible subsets. For this reason, we present the user with preview for only N subsets.

A common challenge that has been addressed in the faceted search literature [Wei et al., 2013] is ranking the facets to display. Wei at el. has identified two major types of facet ranking methods: those that consider each facet independently of others, and those that consider the correlation between facets.

One approach to choosing N subsets independently of others is to pick the top N largest subsets based on their document count. Since this approach ranks each subset independently of others, it is reasonable when the top N subsets do not overlap. However, when there is significant overlap between the chosen subsets, there will be redundant information present in summaries of different sets (i.e., common indicative terms among different subsets). Those redundant summaries may be at the expense of providing summaries that

Table 7.1: Summary generated for the query "*Music*" in the NYT (part 1).

Query	Music
Document count	57,192
Indicative document	Anniversaries Fill the Halls With Melody
Indicative terms	sonata*,orchestra*,album*,concerto*,beethoven*, guitar*,brahm*,quartet*,philharmon*,melodi*, schubert*,mezzo*,bassist*,melod*,saxophonist*, pianist*,symphoni*,saxophon*,bariton*,cello*
Rarely occurring terms	report*
Sub query	Music AND Recordings (Audio)
Document count	7,728
Indicative document	A Once Proud Industry Fends Off Extinction
Indicative terms	cassett*,cd*,reissu*,emi*,disk*,grammophon* ,bmg*,soni*,label*,disc*,rca*, record*,album*,analyst*,nonesuch*,releas*, chart*,parel*,billion*,soundtrack*
Rarely occurring terms	tulli*,pm*,averi*,intermiss*,tomorrow*, ticket*,tonight*,fisher*,carnegi*,saturdai*, hall*,alic*,costum*,metropolitan*,sundai*, recit*,auditorium*,lincoln*,festiv*,concert*
Sub query	Music AND Opera
Document count	6,408
Indicative document	Rossini Hold the Pasta Sauce
Indicative terms	opera*,libretto*,figaro*,puccini*,donizetti*, verdi*,metropolitan*,coloratura*,costum*,wagner*, operat*,luciano*,aria*,rossini*,und*, stage*,scene*,tenor*,la*,dramat*
Rarely occurring terms	saxophonist*,bassist*,guitarist*,rapper*,drummer*, guitar*,album*,songwrit*,band*,sonata*, jam*,rap*,punk*,drum*,jazz*, improvis*,funk*,hop*,saxophon*,rock*

Table 7.2: Summary generated for the query “*Music*” in the NYT (part 2).

Sub query	Music AND Classical Music
Document count	3,580
Indicative document	Bernstein And Bach, Zankel And Zippos
Indicative terms	movement*,intermiss*,beethoven*,mahler*,classic*, bach*,concerto*,shostakovich*,bartok*,cello*, sonata*,schumann*,schoenberg*,liszt*,stravinski*, philharmon*,orchestra*,chopin*,violin*,brahm*
Rarely occurring terms	rapper*,songwrit*,rap*,drummer*,album*, saxophonist*,band*,guitarist*,rock*,blue*, guitar*,drum*,roll*,ballad*,pop*,song*,singer*
Sub query	Music AND Jazz
Document count	2,956
Indicative document	JAZZ
Indicative terms	jazz*,marsali*,theloni*,coltran*,wynton*,bop*, saxophonist*,ellington*,trombonist*,trumpet*,saxophon*, improvis*,alto*,bassist*,drummer*,swing*, afro*,trombon*,cymbal*,gig*
Rarely Occurring Terms	schubert*,brahm*,aria*,haydn*,wagner*,mezzo*, sonata*,verdi*,opera*,mozart*,concerto*,strauss*, beethoven*,philharmon*,choral*,conductor*,symphoni*, cellist*,conduct*,recit*
Sub query	Music AND Rock Music
Document count	2,252
Indicative document	Springsteen: An Old-Fashioned Rocker in a New Era
Indicative terms	rock*,punk*,roll*,beatl*,rocker*,elvi*,band*,fan*, springsteen*,dylan*,mtv*,guitar*,album*,jam*,riff*, catchi*,dj*,keyboardist*,gig*,anthem*
Rarely occurring terms	sonata*,concerto*,brahm*,aria*,schubert*,mozart*, haydn*,soloist*,philharmon*,conductor*,recit*,mezzo*, symphoni*,soprano*,opera*,orchestra*,carnegi*,violinist*, beethoven*,bach*

Table 7.3: Summary generated for the query "*Music*" in the NYT (part 3).

Sub query	Music AND Festivals
Document count	2,594
Indicative document	Some Riffs on a Dream Jazz Festival
Indicative terms	festiv*,tanglewood*,wynton*,marsali*,lincoln*, jazz*,bop*,afro*,averi*,concertgo*, avant*,tribut*,saxophon*,saxophonist*,improvis*, fisher*,trumpet*,featur*,gard*,open*
Rarely occurring terms	emi*,cassett*,cd*,disk*,album*,releas*,record*
Sub query	Music AND Theater
Document count	1,843
Indicative document	CLASSICAL MUSIC AND DANCE GUIDE
Indicative terms	broadwai*,theater*,ballet*,leagu*,theatric*,costum*, dancer*,show*,film*,art*,gershwin*,stage*,danc*, star*,through*,lincoln*,cabaret*,nation*,present*,premier*
Rarely occurring terms	sonata*,chord*,saxophon*,bassist*,improvis*,saxophonist*, textur*,concerto*,guitar*,tempo*,rhythmic*,guitarist*, soloist*,label*,bass*,cd*,phrase*,tone*,drummer*,pianist*
Sub query	Music AND Computers and the Internet
Document count	1,828
Indicative document	EMI to Drop Digital Locks In Web Sales
Indicative terms	analyst*,servic*,protect*,revenu*,billion*, net*,system*,bmg*,law*,court*, compani*,feder*,inc*,chief*,soni*, percent*,disc*,investor*,emi*,cost*
Rarely occurring terms	trio*,quartet*,soprano*,recit*,soloist*, chamber*,concerto*,pm*,tenor*,bariton*, solo*,ballad*,ensembl*,sang*,pianist*, sundai*,symphoni*,carnegi*,violinist*,choru*

Table 7.4: Summary generated for the query "*Music*" in the NYT (part 4).

Sub query	Music AND Motion Pictures
Document count	1,577
Indicative document	Forget the Movie. Listen to the CD
Indicative terms	film*,soundtrack*,senat*,rate*,agenc*, protect*,court*,star*,studio*,democrat*, law*,analyst*,foreign*,billion*,game*, scene*,theater*,million*,govern*,cultur*
Rarely occurring terms	sonata*,quintet*,textur*,virtuos*,tempo*, soloist*,melod*,rhythmic*,brahm*,recit*, quartet*,chord*,solo*,cello*,bariton*, chamber*,concerto*,tenor*,phrase*,drum*
Sub query	Music AND Art
Document count	1,397
Indicative document	Footlights
Indicative terms	art*,ballet*,artist*,film*,through*,theater*, nation*,dancer*,cultur*,celebr*,saturdai*, anniversari*,premier*,open*,modern*,festiv*, danc*,featur*,lincoln*,state*
Rarely occurring terms	riff*,melod*,tempo*,melodi*,disk*, rhythmic*,chord*,sang*,sonata*,phrase*, tone*,label*,harmon*,cd*,ballad*, bass*,guitar*,keyboard*,improvis*,album*

have a broader coverage of the document set of focus. In addition, when the largest subsets are close in size to the set of focus, their summary will provide little additional value because it will be too similar to the general summary.

Similarly, when an indicative set of documents is derived from medoids of overlapping sets, its documents may be too similar to each other and contain redundant information. We expect that medoids that come from two sets where a large number of documents are shared will have a greater similarity to each other than medoids that come from two non-overlapping sets. As a result, the overlap of sets should be considered when choosing the indicative sets.

To ensure that the information scent provides previews for sub-topics that are significant, diverse, and provide good coverage for the set of focus, we need to consider both a subset's size and its similarity to the previously selected subsets. To achieve this we use the *UniformSuggestions* facet value selection algorithm developed by Kashyap et al. [Kashyap et al., 2010], which is designed to select a subset of facet values to present to a user such that the overall expected navigation cost is minimized while at the same time every result is reachable through the presented facet values. The navigation cost is minimized when the selected facet values correspond to subsets of moderate size with minimal overlap. The *UniformSuggestions* is a greedy algorithm that incrementally selects tags that have a high entropy (have moderate selectivity), are popular, and reference many documents that were not accessible through the previously selected tags.

After the subsets are identified, their centroids are derived and used to produce the indicative terms, as explained in Section 7.2, and a set of diverse documents indicative of the contained topics, as explained in Section 7.3.

7.1.2 Subset Coverage

Since subsets are defined in terms of tags, the tagging patterns found in the document collection affect the subsets that are available for each set of focus and how well these subsets cover the documents in it. Subsets $S_{Q \wedge t}$ for all possible values of t, may not be able to cover all the documents found in the document set S_Q.

$$\bigcup_{t \in T} S_{Q \wedge t} \subseteq S_Q \tag{7.1}$$

This problem occurs because documents are assigned varying numbers of tags as is indicated in Figure 3.3. For example, in the NYT collection 34% of documents have a single tag only.

Since for different sets of focus, the corresponding subsets can vary in size and coverage, a different number of subsets may be required for each set of focus in order to provide adequate coverage.

7.1.3 Query Workload

By supporting information scent we change the expected workload processed by the system. Specifically, the system needs to process an increased number of queries, and the average query length is increased.

When a user issues a query to the system, the system needs to generate summaries for $N + 1$ sets of documents; the set of focus and its N subsets. This increase in the number of simultaneous queries issued to the system puts additional strain on it and amplifies the need for fast computation of the centroid measure and the summaries derived from it.

The shift in the query workload to longer queries has an impact on the effectiveness of various partial materialization strategies. For example, if the partial materialization module was implemented using a full cuboid materialization strategy such as the thin cube shell or shell fragments, then in order to support the longer queries, cuboids of higher dimensionality (that consume more space) would need to be materialized by the module, which would result in an increase in the amount of storage space required by the module. In addition, the shell fragment approach assumes that tags are not combined with many other tags. However, to produce information scent we need to perform many such queries, which would result in poor query answering performance.

On the other hand, the individual cell materialization strategies such as MCC do not need any adjustments to support the new query workload and their storage size is unaffected. In addition, their good query performance allows for the larger number of queries to still be answered within a reasonable amount of time. As a result, individual cell materialization strategies are easily extended to support information scent and are an appropriate choice for the partial materialization module.

7.2 Indicative Terms

Given a document set of interest S_c and a superset $S_p \supseteq S_c$ (its context), the *burstiness* of a term w is measured using $B_+(w, C_{S_c}, C_{S_p})$ as defined in Equation 7.2 (which is based on the bursty term measure defined in Equation 2.2 of Section 2.4.2), where C_{S_c} is the centroid of set S_c and C_{S_p} is the centroid of the context set S_p :

$$B_+(w, C_{S_c}, C_{S_p}) = \frac{C_{S_c}[w].\mathit{freq}}{C_{S_p}[w].\mathit{freq}^{0.95}} \tag{7.2}$$

A term is considered *bursty* if it has a sufficiently higher frequency of occurrence in set S_c than its frequency of occurrence in the context set S_p: $B_+(w, C_{S_c}, C_{S_p}) > t$, where t is pre-defined threshold (set to 1 in our system). Since different sets of documents are using different vocabularies, only terms that are present in both vocabularies V_{S_c} and $V_{S_p}$are considered. The top k or fewer (if fewer than k terms are above the threshold) bursty terms found in C_{S_c} based on their burstiness measure constitute the set of indicative terms $\mathbb{B}_k(S_c, S_p)$ for S_c relative to S_p:

$$\mathbb{B}_k(S_c, S_p) = \{w | w \in V_{S_c} \cap V_{S_p} \wedge B_+(w, C_{S_c}, C_{S_p}) > t\}[1 : k] \tag{7.3}$$

Since the absence of terms may often be as informative as their presence, we also want to show terms that occur significantly less in the set than in its superset. Therefore, we define negative burstiness, denoted $B_-(w, C_{S_c}, C_{S_p})$, as defined in Equation 7.4.

$$B_-(w, C_{S_c}, C_{S_p}) = \frac{C_{S_p}[w].\mathit{freq}}{C_{S_c}[w].\mathit{freq}^{0.95}} \tag{7.4}$$

The top k or fewer terms in C_{S_c} with $B_-(w, C_{S_c}, C_{S_p}) > t$ are then displayed as part of the summary, representing a set of rare terms (denoted $\mathbb{B}_k^-(S_c, S_p)$, and defined in Equation 7.5) for S_c relative to S_p.

$$\mathbb{B}_k^-(S_c, S_p) = \{w | w \in V_{S_c} \cap V_{S_p} \wedge B_-(w, C_{S_c}, C_{S_p}) > t\}[1 : k] \tag{7.5}$$

Since both $\mathbb{B}$ and $\mathbb{B}^-$ are derived from the same centroids C_{S_c} and C_{S_p}, no additional retrieval costs are endured to support $\mathbb{B}^-$ in addition to $\mathbb{B}$ measure.

As an example, in Table 7.1 we show a set of positive and negative bursty terms (labeled "indicative terms" and "rarely occurring terms," respectively) for the document set returned in response to the query "*Music*", along with the bursty terms associated with its two largest subsets: "*Music AND Recordings (Audio)*", and "*Music AND Opera*." The indicative terms for the query are derived using $\mathbb{B}_{20}(S_{Music}, S_G)$, where S_G corresponds to all documents in the corpus. While the indicative terms for queries "*Music AND Recordings (Audio)*" and "*Music AND Opera*" are defined with respect to "*Music*" using $\mathbb{B}_{20}(S_{Music \wedge Recordings(Audio)}, S_{Music})$ for the former and $\mathbb{B}_{20}(S_{Music \wedge Opera}, S_{Music})$ for the latter, and the rarely occurring terms using the corresponding negated formulae. The remaining top 10 subsets for query "*Music*" are shown in Tables 7.2, 7.3, and 7.4.

7.3 Indicative Documents

As an additional summary of a collection, we provide the user with a set of diverse documents indicative of the topics in the collection. These documents do not necessarily correspond to a representative sample of the document set, but instead expose the variety of content available. We expect that a set of diverse indicative documents will maximize the amount of information a user gains about the larger collection.

7.3.1 Approaches to Diversification

Using multiple documents to cover all the important concept terms found in the centroid of a set may be appropriate for summarizing a flat set, but in a collection with multi-tagged documents, as is the case with the NYT, most sets can be partitioned into further subsets. For example, a set of documents defined by the "*Music*" tag can be partitioned further into subsets, such as "*Music AND Opera*," derived by intersecting multiple dominant tags found in the set. These subsets impose a hierarchy over the set, which can be used for choosing a set of diverse documents indicative of the contained topics, similarly to what was proposed by Vee et al. [Vee et al., 2008]. A set generated through this approach has documents that represent concepts found in many centroids of various subsets in the set of interest and so will have a richer diversity than is possible with the coverage approach whose documents are optimized to cover a single centroid. This approach parallels the use of query reformulations to provide diverse sets of results for Web searches [Santos et al., 2010].

Instead of calculating the medoid directly from the cosine distance of each document in relation to the centroid, we wish to take advantage of the infrastructure provided by standard information retrieval system such as Lucene. A standard Lucene ranking function, given a query, ranks a document highly if the terms specified in a query appear frequently in the document and the document contains many of the terms specified in the query. Usually, higher weights are assigned by ranking functions to query terms that are rare, but since all bursty terms are relatively rare in the corpus, the ranking function will not likely have a strong bias towards choosing documents that represent only some of the query terms.

The properties used for ranking documents by an IR system align with the properties used for evaluating whether or not a document is a good representative of a set (i.e., it provides good coverage of the important concept terms of the centroid and those terms occur frequently in the document). As a result, the highest ranked document to a query

Table 7.5: Analysis of documents returned for a query on bursty terms with respect to the corresponding tags (where only result sets of size greater than 100 are considered)

Query type	**Instance count**	**Percentage of documents having tags**	**Rank of top document having tags**
t_x	1,015	$62.4_{\sigma=23.0}$	$1.44_{\sigma=1.42}$
$t_x \wedge t_y$	5,179	$70.7_{\sigma=20.6}$	$1.25_{\sigma=0.98}$

consisting of the bursty terms of a set centroid make a good representative document for the corresponding set.

Documents annotated with common tags often share similar terms, which results in a meaningful centroid of a set that contains some dominant concept terms. This may allow for a mapping between tags and text queries, such that when there are m documents with tag t, there exists a query of text terms that will produce a result set whose top m documents are all tagged with tag t. The advantage of using the text query is that it produces a ranked result set whose highest ranked document provides a good representation for the set.

We have examined the quality of this mapping between tags and the top 20 bursty terms describing the corresponding centroids (Table 7.5) to determine whether documents that contain the bursty terms tend to be in the expected subset (having the corresponding tag). The mapping was evaluated over all sets that have at least 100 documents and are defined by a single tag or by a conjunction of two tags. The top 20 bursty terms representing each of those sets were used to formulate queries. For each of those queries, a large proportion of the top 100 documents in the ranked lists of results are annotated with the corresponding tag or tags. Thus, there is a high correlation between the tags and the use of bursty terms in the content text. In addition, the highest ranked result to the query on the bursty terms predominantly comes from the set that was used to generate the bursty terms. Furthermore, since many relevant documents are found in the top 100 results, various IR optimization techniques that take advantage of early termination can be used without fear of missing a representative document.

7.3.2 Finding Indicative Documents

We introduce a new approach for choosing a set of diverse documents indicative of the contained topics for a set of interest in a multi-tagged document collection that uses *Tag-Expansion* as part of its document selection process. Given a query Q defining a set of documents S_Q, and $\mathbb{M}_N$ defining the list of chosen subsets with the *UniformSuggestions* algorithm described in Section 7.1.1, let $S_{m_0}, S_{m_1}, ..., S_{m_n}$ be an ordered sequence of subset found in $\mathbb{M}_N$, and $S_{m_0} = S_Q$. We denote $\mathbb{D}_Q(N, k)$ to be the $N + 1$ diverse documents for S_Q defined using Equation 7.7.

$$D_Q(i,k) = \begin{cases} R(\mathbb{B}_k(S_Q, S_G)) & \text{if } i = 0 \\ R(\mathbb{B}_k(S_{m_i}, S_Q)) & \text{if } i > 0 \end{cases} \tag{7.6}$$

$$\mathbb{D}_Q(N,k) = \bigcup_{i=0}^{N} D_Q(i,k) \tag{7.7}$$

where S_G corresponds to all documents in the corpus and $R(x)$ returns the document in S_Q that is top ranked when posing the text query using the terms in the set x.

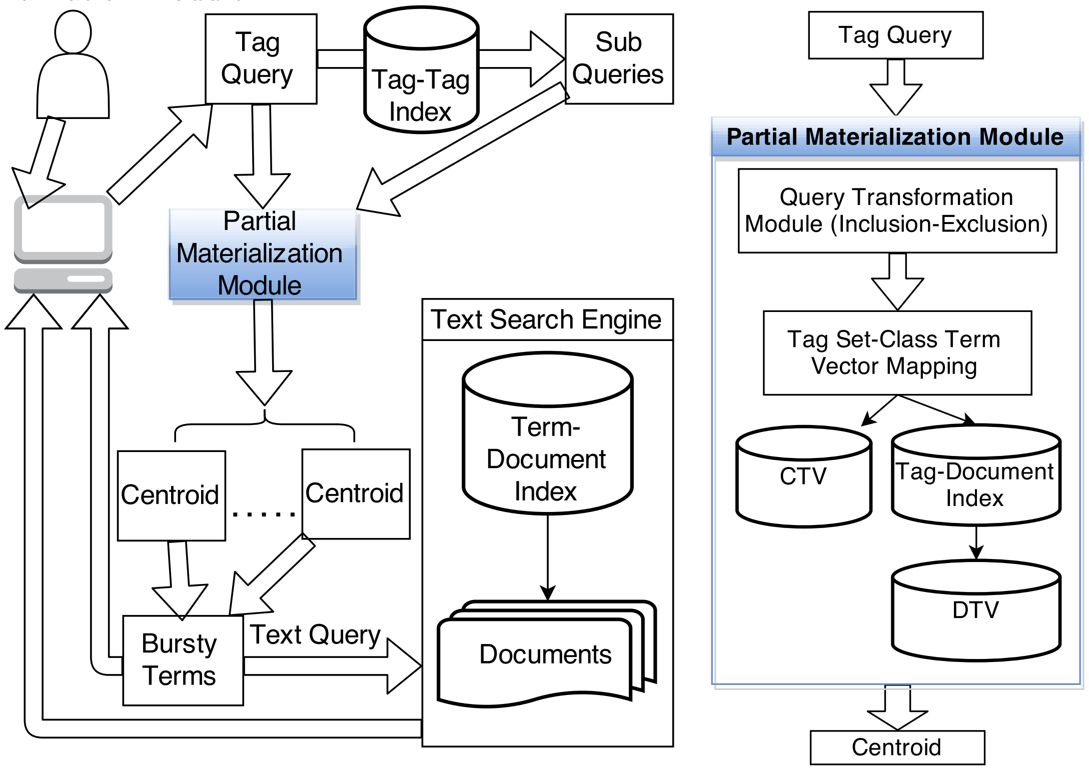
For example, $D_{Music}(0, 20)$ is found by taking the top 20 positive bursty terms, which include stemmed words such as *sonata*, *orchestra*, *album*, *concerto*, and *beethoven*, and issuing a text query with those terms. The highest ranked document in set S_{Music} is chosen as a indicative document, which happens to be the document with title *Anniversaries Fill the Halls With Melody* as shown in Table 7.1. $D_{Music}(1, 20)$ is found using the bursty terms in the largest subset, "*Music AND Recordings (Audio)*," and $D_{Music}(2, 20)$ uses the subset "*Music AND Opera*." For our example, $\mathbb{D}_{Music}(2, 20)$ is displayed in Table 7.1, and these three documents together form a set of diverse documents indicative of the contained topics for S_{Music}.

When the diverse documents are chosen from the N subsets, the cost of computation is small because we benefit from the calculations made to support information scent.

7.4 System Architecture

The architecture of the faceted browsing system is shown in Figure 7.1a. The system is designed around the partial materialization module shown in Figure 7.1b, which is central to efficient system performance and the main topic of this thesis. The partial materialization module is designed to ensure fast access to requested centroids for document sets defined through Boolean operations over tags, while consuming minimal storage. The details of how the module derives a centroid from a Boolean query has been discussed in Section 5.3.3. The fast access to the centroid measure for document sets makes it feasible to produce the proposed summary measures for many sets within a short amount of time, so that a user can interactively explore large document collections.

Figure 7.1: a) System for supporting faceted browsing and b) zoomed in view of partial materialization module



The system produces summaries for documents corresponding to a query over tags specified by a user by first accessing a Tag-Tag Index along with a Tag-Document Index (found in the Partial Materialization Module) to determine the indicative subsets that will be used to produce the summary for the document set of focus. For the query "*Music*", sub-queries such as "*Music AND Recordings (Audio)*" and "*Music AND Opera*" are generated. The original query along with the N sub-queries are then fed into the partial materialization module, which produces a centroid for each of the queries along with the centroid for the whole corpus.

The resulting centroids are compared against each other to derive both positive and negative bursty terms (Eqns. 7.2 and 7.4) for the result set and subsets of focus. The positive bursty terms are then used to form multiple text queries (a separate query for each set of bursty terms) (Eqns. 7.6 and 7.7). The highest ranked documents returned from each search along with the corresponding bursty terms are returned to the user to produce the summary of the resulting set.

The user interface is developed as a Web app, and Lucene is used for indexing the documents and for supporting text queries through the standard ranking function.

7.5 Evaluation

7.5.1 Time Analysis for Retrieving Centroids

To give more clarity about the savings in query execution costs that is achieved by using our materialization strategies, we measured the time required to generate the requested centroid by a system that implements no materialization (NM) and one that implements the MCC materialization strategy.[1] We measured the performance of the two systems on seven conjunctive queries over the NYT collection reflecting various set sizes as shown in Table 7.6. The time required to answer a query using NM depends on the number of documents that need to be aggregated and generally increases with the number of documents that need to be aggregated. Surprisingly, it takes more time to answer the query 'Music' than the query 'Finances', which has 2.5 times more documents. We suspect that this might be because of how the documents are clustered on the disk. The time required to answer a query when MCC materialization is used depends on the number of blocks that need to be retrieved as well as the number of additional documents that need

[1]The time required for answering each query was measured on a cold start to ensure that no data was preloaded into memory.

Table 7.6: Time comparison between MCC and NM

Query	**Doc. count**	**Blocks retr.**	**Docs retr.**	**Time (ms) MCC**	**NM**	**NM/ MCC**
Finances	142,609	1	0	1,311	122,237	93.24
Music	57,192	1	0	1,191	136,608	114.70
Dancing	13,798	1	0	1,168	71,735	61.42
Music AND Opera	6,408	1	0	1,284	60,096	46.80
Health Insurance and Managed Care AND Prices (Fares, Fees and Rates)	658	1	49	3,329	11,026	3.31
Finances AND Personal Finances AND Bankruptcies	50	1	0	1,389	3,085	2.22
Design AND Auctions	49	0	49	3,457	3,243	0.94

to be retrieved to adjust the retrieved block to correspond to the appropriate set. As these seven queries demonstrate on a real system, the time required to produce the answer when using MCC is bounded and takes a fraction of time required by NM. These results are similar to what was shown in Section 5.4.2.

7.5.2 Approaches to Picking Diverse Documents

We evaluate the utility of the proposed *TagExpansion* approach to selecting a set of indicative documents by comparing it against two alternative techniques that have been previously proposed in the literature: *ClusterMedoid* and *ClusterPrinDocs*.

The *ClusterMedoid*[2] approach to selecting diverse set of documents against which we compare our *TagExpansion* approach has been used in systems such as the Scatter/Gather [Cutting et al., 1992], where a set of documents S_Q is clustered using k-means clustering into N disjoint clusters based on terms found in C_Q. A medoid document is then taken from each cluster to form a set of diverse documents indicative of the produced clusters. Since clustering is performed online, this method is slow when $|S_Q|$ is large because retrieving a large number of documents from the disk and then performing clustering on them is time consuming.

The implementation of *ClusterPrinDocs* [Deolalikar, 2014] produces a set of up to N documents used for summarizing a document set S_Q by picking documents that together

[2]The *ClusterMedoid* name is only used for the sake of distinction between the other algorithms that are being compared, and has not been labeled this way in the Scatter/Gather literature.

cover all the terms found in the centroid C_Q. The algorithm relies on a coverage-profile P that, for each term in the centroid, specifies the amount of occurrence required by a term in the selected set of documents in order to be considered covered. As part of our evaluation, the values of the coverage-profile are set to the mean term frequencies of the centroid. The documents used for summarizing a set are picked by ranking the documents in the set by how well they cover the uncovered terms in the coverage-profile. The documents that are ranked highest are picked as indicative documents and the coverage profile is updated to account for the terms covered by the newly picked documents. The updated set of uncovered terms in the coverage profile is then used to re-rank the documents so that the next highest ranked document can be picked. This process continues until the coverage-profile is fully covered or N documents are picked. In our implementation of the algorithm, we have used the uncovered set of terms from the coverage profile to issue queries and relied on Lucene's inverted index and scoring function to pick the indicative documents in the same manner as we have done with our *TagExpansion* approach. (The *ClusterPrinDocs* algorithm in its original form uses an inefficient ranking implementation that requires all the set member documents to be retrieved from disk before the ranking can be performed.)

Since both the *TagExpansion* and *ClusterPrinDocs* approaches require access to centroid data of the underlying set in order to derive the indicative documents, they both benefit from the fast access to centroid data provided by the partial materialization module developed in this thesis.

Both the *TagExpansion* and *ClusterPrinDocs* algorithms are iterative and their selection process of the next indicative document is independent of the size N and so $\mathbb{D}_Q(N-1) \subseteq \mathbb{D}_Q(N)$. On the other hand, the *ClusterMedoid* approach produces a completely different set of indicative documents for different values of N.

To provide a frame of reference for evaluating the performance of the three above approaches, we consider a fourth approach towards selecting a diverse set of documents, which selects a random set of N documents with a uniform probability.

All the approaches are evaluated on 1,006 single tag queries that produce result sets ranging in size between 201 and 46,246 documents. We examine sets of indicative documents of size 10 and 20.

7.5.3 Measures Used

There are various measures used in the search result diversification literature for quantifying the quality of the result set [Santos et al., 2015]. However, these measures deal with evaluating the quality of a diversified search result ranking for underspecified queries, where each underspecified query is assumed to address multiple aspects of an information need and the diversified result sets are evaluated based on how many different aspects of the information need they cover and how good they are at representing each aspect. Since in our situation we do not have queries or access to aspects (unless one would treat each tag as an aspect), these measures cannot be applied to our scenario. As a result, we introduce a set of measures that are more appropriate to our situation.

The indicative document sets produced by the above four approaches are examined on seven measures in order to get insight about the properties of the results produced by them. The seven measures include time, average minimum document distance, average document distance, term coverage, weighted term coverage, tag coverage, and weighted tag coverage. In all cases, the reported performance of the random approach is the mean of 100 random samples. All measures that rely on terms (average minimum document distance, average document distance, term coverage, and weighted term coverage) use the top 500 features in the set of focus based on mutual information. The details of each of the measures are defined as follow:

- **Time** measures the time, in seconds, that is required to compute the set of indicative documents. The time consumed by the *ClusterPrinDocs* depends on the cost of answering multiple keyword queries and the cost of retrieving the DTVs of all the indicative document. For the *TagExpansion*, since the indicative documents are picked by relying on the same set of centroids as are used for supporting information scent, the time consumed is only affected by the cost of issuing N queries of length 20 (number of bursty terms used in queries). For the *ClusterMedoid*, since it performs clustering on individual documents, the time cost is affected by the time required to retrieve all the DTV's and then clustering them.

- **Average Minimum Document Distance** is the average minimum cosine distance between the DTVs of the documents found in the set of indicative documents. Since many features are projected out, it is possible for some documents to have zero remaining features (which results in a maximum distance between it and all other documents).

- **Average Document Distance** is the average cosine distance between DTVs of the documents found in the set of indicative documents.

- **Term Coverage** examines what percentage of the 500 features in the set of focus are present in the set of indicative documents.

Table 7.7: Analysis on diverse indicative documents chosen by four different strategies on 1,006 single tag queries. Ten documents are chosen per set and 500 features are used for evaluating distances between documents and centroids.

Measure	**Algorithm**			
	Tag Expansion	*Cluster PrinDocs*	*Cluster Medoid*	Random
Time (s)	2.57 $_{\sigma=2.21}$	*2.86 $_{\sigma=3.90}$	24.94 $_{\sigma=72.01}$	
Avg Min Doc Dist	0.45 $_{\sigma=0.08}$	0.42 $_{\sigma=0.08}$	0.42 $_{\sigma=0.08}$	0.50 $_{\sigma=0.06}$
Avg Doc Dist	0.66 $_{\sigma=0.08}$	0.59 $_{\sigma=0.09}$	0.60 $_{\sigma=0.09}$	0.69 $_{\sigma=0.06}$
Term Coverage	0.66 $_{\sigma=0.12}$	0.85 $_{\sigma=0.09}$	0.71 $_{\sigma=0.11}$	0.56 $_{\sigma=0.11}$
Term Weighted Coverage	0.92 $_{\sigma=0.04}$	0.98 $_{\sigma=0.01}$	0.95 $_{\sigma=0.03}$	0.89 $_{\sigma=0.04}$
Tag Coverage	0.10 $_{\sigma=0.07}$	0.07 $_{\sigma=0.05}$	0.09 $_{\sigma=0.05}$	0.08 $_{\sigma=0.04}$
Tag Weighted Coverage	0.56 $_{\sigma=0.16}$	0.53 $_{\sigma=0.16}$	0.54 $_{\sigma=0.16}$	0.50 $_{\sigma=0.16}$

- **Weighted Term Coverage** is similar to the term coverage measure, except that each term has a weight assigned equal to its frequency in the centroid. The measure is normalized by dividing it by the sum of all the term frequencies found in the centroid.

- **Tag Coverage** is the percentage of all distinct tags that are found in the set of focus that are present in the indicative set of documents. This measure is equivalent to the subtopic recall at rank k, introduced by Zhai et al. [Zhai et al., 2003] to measure the percentage of subtopics covered by the top k documents.

- **Weighted Tag Coverage** is similar to the tag coverage measure, except that the weight of each tag is equal to the ratio of documents in the set of focus that have the tag. The score is normalized by dividing by the maximum achievable score.

7.5.4 Observations

The properties of the sets of diverse documents generated by the four approaches are shown in Table 7.7 for sets of size 10, and in Table 7.8 for sets of size 20. For some sets of focus, the *ClusterPrinDocs* can produces less than the pre-specified number of indicative documents because the coverage-profile is fully covered with fewer documents. Similarly, some sets of focus have fewer subsets than the pre-specified number of requested documents, which

Table 7.8: Analysis on diverse indicative documents chosen by four different strategies on 1,006 single tag queries. Twenty documents are chosen per set and 500 features are used for evaluating distances between documents and centroids.

Measure	**Algorithm**			
	Tag Expansion	*Cluster PrinDocs*	*Cluster Medoid*	Random
Time (s)	$2.93_{\sigma=3.55}$	$3.18_{\sigma=3.13}$	$22.51_{\sigma=39.00}$	
Avg Min Doc Dist	$0.42_{\sigma=0.06}$	$0.41_{\sigma=0.06}$	$0.41_{\sigma=0.07}$	$0.46_{\sigma=0.05}$
Avg Doc Dist	$0.67_{\sigma=0.07}$	$0.63_{\sigma=0.08}$	$0.64_{\sigma=0.08}$	$0.69_{\sigma=0.06}$
Term Coverage	$0.78_{\sigma=0.11}$	$0.94_{\sigma=0.07}$	$0.81_{\sigma=0.09}$	$0.70_{\sigma=0.10}$
Term Weighted Coverage	$0.96_{\sigma=0.03}$	$0.99_{\sigma=0.01}$	$0.98_{\sigma=0.01}$	$0.95_{\sigma=0.02}$
Tag Coverage	$0.15_{\sigma=0.09}$	$0.12_{\sigma=0.07}$	$0.14_{\sigma=0.08}$	$0.12_{\sigma=0.06}$
Tag Weighted Coverage	$0.67_{\sigma=0.12}$	$0.62_{\sigma=0.14}$	$0.64_{\sigma=0.14}$	$0.60_{\sigma=0.14}$

causes the *TagExpansion* to produce smaller sets. In such situations, the four approaches are evaluated on the maximum number of documents that can be achieved by all.

Since many of the observed results for the four algorithms appear close to each other, we have performed a Wilcoxon signed-rank test [Candan and Sapino, 2010] on all the results, comparing *TagExpansion* to the other approaches. Based on this test, all but the difference between *TagExpansion* and *ClusterPrinDocs* for "time" in Table 7.7 are statistically significant.

The time measure highlights the main drawback of the *ClusterMedoid* approach, which makes it infeasible to be used in our system. As the size of the document set of focus grows, it takes unacceptably long to compute the corresponding set of indicative documents. In contrast, the *TagExpansion*, takes the least amount of time to compute the indicative documents. This short computation time is achieved by reusing the same bursty terms that are used to support information scent. However, if the set of indicative documents is derived from different subsets than the ones used for supporting information scent, then the required computation time will be considerably higher. Additionally, with the *TagExpansion*, all N queries are known in advance and can be issued simultaneously and parallelized. This is not possible with *ClusterPrinDocs*, which needs to retrieve the DTVs of the best documents for each query, in order to update the coverage-profile and determine what is the next query that it needs to issue. In addition, the initial keyword queries generated by the *ClusterPrinDocs* are long (500 terms in our experiments), and in turn, more expensive to compute than then the queries generated by *TagExpansion* (20 terms).

Compared to *ClusterPrinDocs* and *ClusterMedoid*, *TagExpansion* has the highest average minimum document distance and highest average distance between indicative documents, which suggests its documents are the most diverse out of the three. Randomly chosen documents have a larger average minimum and average document distance than *TagExpansion*, but the reason for this is that the randomly chosen documents often have none or very few of the desired features, as can be seen from the low value for the term coverage. This suggests that randomly chosen documents are more diverse but less indicative of the important concepts of the set. The *TagExpansion* algorithm, on the other hand, maintains a high average minimum document distance while also maintaining a high term coverage.

For the term coverage, the *ClusterPrinDocs* performs the best, which is expected since it is designed to maximize the coverage of the centroid. *TagExpansion*, instead, searches for documents that have *bursty terms*, which are different from the actual features used in the set of focus. The *TagExpansion* performs better than the other algorithms on the measures that evaluate the indicative document sets based on tag coverage since it is designed to provide diversity on tags.

Since in a faceted browser we expect the user to navigate through a selection of tags and we assume that the tags are of high quality, we expect it is more desirable to have the documents provide a good representation of the tags found in the result set. As a result, the *TagExpansion* approach appears to be an appropriate choice for selecting a diverse set of indicative documents.

7.6 Conclusions

Whereas a standard faceted browser in response to a user's query on a tag "*Music*" would produce a list of all documents that have that tag (which happen to be 57,192 documents), as well as a list of subsets and the number of documents within each subset ("*Recordings (Audio)*" with 7,728 documents, "*Opera*" with 6,408 documents, etc.), our system, in addition, provides a set of diverse documents that come from the various subsets (defined by queries "*Music AND Recordings (Audio)*", "*Music AND Opera*", etc.) as well as indicative terms for the documents labeled with "*Music.*" In addition, our system provides a set of indicative terms for the subsets, to provide an information scent for the user that can inform their navigation decision as to which subset they may want to explore further.

Chapter 8

Conclusions and Future Directions

Our work has been motivated by the goal of enabling users to explore large multi-tagged document collections efficiently. To this end, we aimed to enrich the capabilities of faceted browsers through the addition of result set summarization, so that users can explore and quickly understand the contents of a large collection. To achieve this goal, the challenge was to develop a system that, in response to a query on tags (that defines a set of documents), quickly produces a rich and accurate summary. The system was built under the constraint of limited storage space, as well as a need to be robust and capable of handling different collections. The solution developed in this thesis addresses all of these challenges.

To produce summary measures in a short time while consuming minimal space, we took inspiration from OLAP, where partial materialization strategies are used to address these challenges. However, multi-tagged document collections contain unique properties that prohibit the reuse of previously developed materialization strategies. As a result, we performed a detailed analysis of the tagging patterns found in the New York Times and the ACM Digital Library, based on which we developed a novel individual cell materialization strategy and various partial materialization strategies that work on top of it. We compared our partial materialization strategies to competing approaches and demonstrated that our techniques consume less space and offer faster response time on expected queries.

Furthermore, based on the NYT and ACM collections, we developed a generative tagging model that mimics the tagging patterns observed in the two collections. This generative model was then used to generate a wide range of synthetic collections with various properties, which, along with the held-back PubMed collection, were used to demonstrate the robustness of our approach to a variety of collections.

To support rich summaries while consuming minimum space, we analyzed the distribution of terms inside document collections. Since document collections tend to have very large vocabularies that are not feasible to be stored in full, we reduced the size of vocabulary through feature selection and evaluated multiple vocabulary storage approaches. Based on our analysis, we have developed a local and an enriched-local vocabulary storage approach that provide a much richer vocabulary than the traditional global storage approach for summarizing document sets while consuming the same amount of space. We have then showed how those vocabulary storage techniques can be incorporated into the proposed partial materialization and developed an algorithm for efficient materialization.

Finally, an enhanced faceted browsing system was built around the developed partial materialization module to demonstrate the benefits of having the document set summaries available to the user and accessible in an interactive manner. The system was designed to provide information scent and summaries for result sets through the use of indicative terms and a diverse set of indicative documents. These are calculated by leveraging the fast access to centroids for sets of interest and their subsets. The set of diverse indicative documents is chosen through a novel approach that transforms a tag query to a text query and takes advantage of the infrastructure provided by standard information retrieval systems.

As part of our work we have combined the areas of OLAP, partial materialization, information retrieval, topic modeling, feature selection, cluster labelling, text summarization, and faceted browsing in a unique way to build a complete solution that can be applied to many different multi-tagged document collections and help in their exploration, analysis, and knowledge discovery. In the remainder of this chapter, we indicate some of the many directions in which this research can be continued.

In our work we have made many assumptions about how users would explore a collection and the type of queries they would pose. We have relied on the PubMed query workload for modeling the type and length of expected queries. We also made assumptions about what makes a good indicative document. To confirm that our assumptions reflect real world usage, it would be useful to perform user studies.

The NYT and ACM collections motivated the need for picking a set of indicative documents for summarizing a result set by taking advantage of the tagging patterns. However, by not having access to the ground truth, we were unable to evaluate if, in fact, any of the approaches produced good results. It would be beneficial to find a suitable test collection on which those approaches could be evaluated.

We have developed an approach that picks a set of indicative documents by leveraging fast access to bursty terms. However, further analysis of this approach should be performed to determine what is an appropriate scoring function to use for ranking the queries on bursty terms.

In the current implementation, we determine the bursty terms for a subset by comparing the frequencies of the terms against their frequencies in the superset. The identified bursty terms are then used to generate a query. However, a different set of bursty terms would be obtained if the frequencies of the terms in the subset were compared against the global frequencies. This, in turn, would result in a different query and (presumably) a different indicative document. Does one approach perform better than the other?

Currently we have summarized sets of documents through indicative documents that correspond to documents that contain bursty terms. In a similar fashion, it would be interesting to investigate if documents that have a high occurrence of terms that are rare in the set provide any useful information to the user. Do those documents present unexpected information that would be of value and are analogues to an outlier?

In the current materialization design, we did not consider the temporal dimension. Since most documents are annotated with the date of creation, incorporating time into the design would be an interesting next step.

As part of the current analysis of tagging patterns in the document collections, we only considered a flat vocabulary of tags; the hierarchical properties of tags were ignored. For example, the category tags in the ACM, such as *C.2.1:Wireless communication* and *C.2*, were treated as completely independent. The location of tags in the taxonomy was ignored. If the vocabulary of tags is hierarchical, then corresponding operations of "roll-up" and "drill-down" within the hierarchies should be supported. It would be interesting to investigate if there is any value and associated challenges to supporting a richer tag taxonomy.

In the current design we have not considered updates, which could be caused by tags added to documents or new documents added to the collection. Examining how to have the partial materialization system handle such updates incrementally without rebuilding the materialization may be an interesting extension to the research.

We have shown in this thesis that it is possible to have a materialization strategy that can support queries over tags. We have also shown that there is some alignment between the results that are produced to a text query on bursty terms and a query posed on tags. Based on this observation, it would be interesting to investigate if the aggregation of the result set produced by a keyword search could benefit from using the materialized cells. If the queries posed by users are composed of what we refer to as bursty terms, then there is a possibility that we could take advantage of the materialized cells. To test this hypothesis we would need access to query logs.

Currently, we have only applied the system to help explore document collections. However, it would be interesting to apply the system to help with exploration and summarization of review websites, such as hotel reviews, restaurant reviews, product reviews, and Google Play reviews. Review websites have much redundant text since there are a lot of individuals commenting on the same items, and individuals reading reviews could greatly benefit from the summarization.

Appendix

Appendix A

List of all Queries Involving 3 Tags

In Table A.1, we list all the possible query patterns that can be performed with three tags. The query patterns are in their shortest form, since we expect the user to issue shortest queries for expressing their intent. The queries are in ascending order based on the number of operators used. For each query, we provide the number of query combinations that use the specific pattern. In addition, we provide the cost and the cells that need to be aggregated when the query is answered using full cuboid materialization strategy as well as individual cell materialization strategy. All together, there are 127 possible queries, which are grouped into 39 different query patterns.

The query patterns annotated with an asterisk do not contain the negation operation. They produce the 18 queries output by the query workload generator that is derived from the analysis of the PubMed query log, as is described in Section 5.2.3.

Table A.1: List of all 127 queries that can be performed with three tags and the corresponding cost associated with answering them when relying on cuboid materialization and individual cell materialization.

Id	Query			Cost		Cuboid	I(T)
	Len	Pattern	#	3-D	I(T)	aggregation	aggregation
*1	1	t_1	3	4	1	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{t_1\wedge t_2\wedge\neg t_3}$ $+C_{t_1\wedge\neg t_2\wedge t_3} + C_{t_1\wedge t_2\wedge t_3}$	C_{t_1}
*2	2	$t_1 \wedge t_2$	3	2	1	$C_{t_1\wedge t_2\wedge\neg t_3} + C_{t_1\wedge t_2\wedge t_3}$	$C_{t_1\wedge t_2}$
3	2	$t_1 \wedge \neg t_2$	6	2	2	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{t_1\wedge\neg t_2\wedge t_3}$	$C_{t_1} - C_{t_1\wedge t_2}$
*4	2	$t_1 \vee t_2$	3	6	3	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{\neg t_1\wedge t_2\wedge\neg t_3}$ $+C_{t_1\wedge t_2\wedge\neg t_3} + C_{t_1\wedge\neg t_2\wedge t_3}$ $+C_{\neg t_1\wedge t_2\wedge t_3} + C_{t_1\wedge t_2\wedge t_3}$	$C_{t_1} + C_{t_2} - C_{t_1\wedge t_2}$
*5	3	$t_1 \wedge t_2 \wedge t_3$	1	1	1	$C_{t_1\wedge t_2\wedge t_3}$	$C_{t_1\wedge t_2\wedge t_3}$
6	3	$t_1 \wedge t_2 \wedge \neg t_3$	3	1	2	$C_{t_1\wedge t_2\wedge\neg t_3}$	$C_{t_1\wedge t_2} - C_{t_1\wedge t_2\wedge t_3}$
7	3	$t_1\wedge$ $(\neg t_2 \vee \neg t_3)$	3	3	2	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{t_1\wedge t_2\wedge\neg t_3}$ $+C_{t_1\wedge\neg t_2\wedge t_3}$	$C_{t_1} - C_{t_1\wedge t_2\wedge t_3}$
*8	3	$t_1 \wedge (t_2 \vee t_3)$	3	3	3	$C_{t_1\wedge t_2\wedge\neg t_3} + C_{t_1\wedge\neg t_2\wedge t_3}$ $+C_{t_1\wedge t_2\wedge t_3}$	$C_{t_1\wedge t_2} + C_{t_1\wedge t_3}$ $-C_{t_1\wedge t_2\wedge t_3}$
9	3	$t_1 \wedge (\neg t_2 \vee t_3)$	6	3	3	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{t_1\wedge\neg t_2\wedge t_3}$ $+C_{t_1\wedge t_2\wedge t_3}$	$C_{t_1} - C_{t_1\wedge t_2}$ $+C_{t_1\wedge t_2\wedge t_3}$
*10	3	$t_1 \vee t_2 \wedge t_3$	3	5	3	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{t_1\wedge t_2\wedge\neg t_3}$ $+C_{t_1\wedge\neg t_2\wedge t_3} + C_{\neg t_1\wedge t_2\wedge t_3}$ $+C_{t_1\wedge t_2\wedge t_3}$	$C_{t_1} + C_{t_2\wedge t_3}$ $-C_{t_1\wedge t_2\wedge t_3}$
11	3	$t_1 \wedge \neg t_2 \wedge \neg t_3$	3	1	4	$C_{t_1\wedge\neg t_2\wedge\neg t_3}$	$C_{t_1} - C_{t_1\wedge t_2}$ $-C_{t_1\wedge t_3} + C_{t_1\wedge t_2\wedge t_3}$
12	3	$t_1 \wedge \neg t_3 \vee t_2$	6	5	5	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{\neg t_1\wedge t_2\wedge\neg t_3}$ $+C_{t_1\wedge t_2\wedge\neg t_3} + C_{\neg t_1\wedge t_2\wedge t_3}$ $+C_{t_1\wedge t_2\wedge t_3}$	$C_{t_1} + C_{t_2} - C_{t_1\wedge t_2}$ $-C_{t_1\wedge t_3} - C_{t_1\wedge t_2\wedge t_3}$
13	3	$(t_1 \vee t_2) \wedge \neg t_3$	3	3	6	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{\neg t_1\wedge t_2\wedge\neg t_3}$ $+C_{t_1\wedge t_2\wedge\neg t_3}$	$C_{t_1} + C_{t_2} - C_{t_1\wedge t_2}$ $-C_{t_1\wedge t_3} - C_{t_2\wedge t_3}$ $+C_{t_1\wedge t_2\wedge t_3}$
*14	3	$t_1 \vee t_2 \vee t_3$	1	7	7	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{\neg t_1\wedge t_2\wedge\neg t_3}$ $+C_{\neg t_1\wedge\neg t_2\wedge t_3} + C_{t_1\wedge t_2\wedge\neg t_3}$ $+C_{t_1\wedge\neg t_2\wedge t_3} + C_{\neg t_1\wedge t_2\wedge t_3}$ $+C_{t_1\wedge t_2\wedge t_3}$	$C_{t_1} + C_{t_2} + C_{t_3}$ $-C_{t_1\wedge t_2} - C_{t_1\wedge t_3}$ $-C_{t_2\wedge t_3} + C_{t_1\wedge t_2\wedge t_3}$

Id	Query			Cost		Cuboid	I(T)
	Len	Pattern	#	3-D	I(T)	aggregation	aggregation
15	4	$t_1 \wedge \neg t_2$ $\vee \neg t_1 \wedge t_2$	3	4	3	$C_{t_1 \wedge \neg t_2 \wedge \neg t_3} + C_{\neg t_1 \wedge t_2 \wedge \neg t_3}$ $+ C_{t_1 \wedge \neg t_2 \wedge t_3} + C_{\neg t_1 \wedge t_2 \wedge t_3}$	$C_{t_1} + C_{t_2} - 2C_{t_1 \wedge t_2}$
16	4	$t_1 \wedge \neg t_2 \vee$ $\vee t_2 \wedge t_3$	6	4	3	$C_{t_1 \wedge \neg t_2 \wedge \neg t_3} + C_{t_1 \wedge \neg t_2 \wedge t_3}$ $+ C_{\neg t_1 \wedge t_2 \wedge t_3} + C_{t_1 \wedge t_2 \wedge t_3}$	$C_{t_1} - C_{t_1 \wedge t_2} + C_{t_2 \wedge t_3}$
17	4	$t_1 \wedge \neg t_3$ $\vee \neg t_1 \wedge t_2$	6	4	4	$C_{t_1 \wedge \neg t_2 \wedge \neg t_3} + C_{\neg t_1 \wedge t_2 \wedge \neg t_3}$ $+ C_{t_1 \wedge t_2 \wedge \neg t_3} + C_{\neg t_1 \wedge t_2 \wedge t_3}$	$C_{t_1} + C_{t_2} - C_{t_1 \wedge t_2}$ $- C_{t_1 \wedge t_3}$
18	5	$t_1 \wedge (t_2 \wedge \neg t_3$ $\vee \neg t_2 \wedge t_3)$	3	2	3	$C_{t_1 \wedge t_2 \wedge \neg t_3} + C_{t_1 \wedge \neg t_2 \wedge t_3}$	$C_{t_1 \wedge t_2} + C_{t_1 \wedge t_3}$ $- 2C_{t_1 \wedge t_2 \wedge t_3}$
19	5	$t_1 \wedge (t_2 \wedge t_3$ $\vee \neg t_2 \wedge \neg t_3)$	3	2	4	$C_{t_1 \wedge \neg t_2 \wedge \neg t_3} + C_{t_1 \wedge t_2 \wedge t_3}$	$C_{t_1} - C_{t_1 \wedge t_2} - C_{t_1 \wedge t_3}$ $+ 2C_{t_1 \wedge t_2 \wedge t_3}$
20	5	$t_1 \wedge \neg t_2$ $\vee \neg t_1 \wedge t_2 \wedge t_3$	6	3	4	$C_{t_1 \wedge \neg t_2 \wedge \neg t_3} + C_{t_1 \wedge \neg t_2 \wedge t_3}$ $+ C_{\neg t_1 \wedge t_2 \wedge t_3}$	$C_{t_1} - C_{t_1 \wedge t_2} + C_{t_2 \wedge t_3}$ $- C_{t_1 \wedge t_2 \wedge t_3}$
*21	5	$t_1 \wedge (t_2 \vee t_3)$ $\vee t_2 \wedge t_3$	1	4	4	$C_{t_1 \wedge t_2 \wedge \neg t_3} + C_{t_1 \wedge \neg t_2 \wedge t_3}$ $+ C_{\neg t_1 \wedge t_2 \wedge t_3} + C_{t_1 \wedge t_2 \wedge t_3}$	$C_{t_1 \wedge t_2} + C_{t_1 \wedge t_3}$ $+ C_{t_2 \wedge t_3} - 2C_{t_1 \wedge t_2 \wedge t_3}$
22	5	$t_1 \wedge \neg t_2$ $\vee t_2 \wedge$ $(\neg t_1 \vee t_3)$	3	5	4	$C_{t_1 \wedge \neg t_2 \wedge \neg t_3} + C_{\neg t_1 \wedge t_2 \wedge \neg t_3}$ $+ C_{t_1 \wedge \neg t_2 \wedge t_3} + C_{\neg t_1 \wedge t_2 \wedge t_3}$ $+ C_{t_1 \wedge t_2 \wedge t_3}$	$C_{t_1} + C_{t_2}$ $- 2C_{t_1 \wedge t_2} + C_{t_1 \wedge t_2 \wedge t_3}$
23	5	$t_1 \wedge$ $(\neg t_2 \vee \neg t_3)$ $\vee \neg t_1 \wedge t_2$	3	5	4	$C_{t_1 \wedge \neg t_2 \wedge \neg t_3} + C_{\neg t_1 \wedge t_2 \wedge \neg t_3}$ $+ C_{t_1 \wedge t_2 \wedge \neg t_3} + C_{t_1 \wedge \neg t_2 \wedge t_3}$ $+ C_{\neg t_1 \wedge t_2 \wedge t_3}$	$C_{t_1} + C_{t_2}$ $- C_{t_1 \wedge t_2} - C_{t_1 \wedge t_2 \wedge t_3}$
24	5	$t_1 \wedge \neg t_2 \wedge \neg t_3$ $\vee \neg t_1 \wedge t_2$	6	3	5	$C_{t_1 \wedge \neg t_2 \wedge \neg t_3} + C_{\neg t_1 \wedge t_2 \wedge \neg t_3}$ $+ C_{\neg t_1 \wedge t_2 \wedge t_3}$	$C_{t_1} + C_{t_2} - 2C_{t_1 \wedge t_2}$ $- C_{t_1 \wedge t_3} + C_{t_1 \wedge t_2 \wedge t_3}$
25	5	$t_1 \wedge \neg t_2 \wedge \neg t_3$ $\vee t_2 \wedge t_3$	3	3	5	$C_{t_1 \wedge \neg t_2 \wedge \neg t_3} + C_{\neg t_1 \wedge t_2 \wedge t_3}$ $+ C_{t_1 \wedge t_2 \wedge t_3}$	$C_{t_1} - C_{t_1 \wedge t_2} - C_{t_1 \wedge t_3}$ $+ C_{t_2 \wedge t_3} + C_{t_1 \wedge t_2 \wedge t_3}$
26	5	$\neg t_3 \wedge (t_1 \wedge \neg t_2$ $\vee \neg t_1 \wedge t_2)$	3	2	6	$C_{t_1 \wedge \neg t_2 \wedge \neg t_3} + C_{\neg t_1 \wedge t_2 \wedge \neg t_3}$	$C_{t_1} + C_{t_2} - 2C_{t_1 \wedge t_2}$ $- C_{t_1 \wedge t_3} - C_{t_2 \wedge t_3}$ $+ 2C_{t_1 \wedge t_2 \wedge t_3}$
27	5	$t_1 \wedge (t_2 \vee \neg t_3)$ $\vee t_2 \wedge \neg t_3$	3	4	6	$C_{t_1 \wedge \neg t_2 \wedge \neg t_3} + C_{\neg t_1 \wedge t_2 \wedge \neg t_3}$ $+ C_{t_1 \wedge t_2 \wedge \neg t_3} + C_{t_1 \wedge t_2 \wedge t_3}$	$C_{t_1} + C_{t_2} - C_{t_1 \wedge t_2}$ $- C_{t_1 \wedge t_3} - C_{t_2 \wedge t_3}$ $+ 2C_{t_1 \wedge t_2 \wedge t_3}$
28	5	$t_1 \wedge \neg t_2$ $\vee \neg t_1 \wedge (t_2 \vee t_3)$	3	5	7	$C_{t_1 \wedge \neg t_2 \wedge \neg t_3} + C_{\neg t_1 \wedge t_2 \wedge \neg t_3}$ $+ C_{\neg t_1 \wedge \neg t_2 \wedge t_3} + C_{t_1 \wedge \neg t_2 \wedge t_3}$ $+ C_{\neg t_1 \wedge t_2 \wedge t_3}$	$C_{t_1} + C_{t_2} + C_{t_3}$ $- 2C_{t_1 \wedge t_2} - C_{t_1 \wedge t_3}$ $- C_{t_2 \wedge t_3} + C_{t_1 \wedge t_2 \wedge t_3}$

Id	Query			Cost		Cuboid	I(T)
	Len	Pattern	#	3-D	I(T)	aggregation	aggregation
29	5	$t_1 \wedge \neg t_2 \vee \neg t_1 \wedge t_2 \vee t_3$	3	6	7	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{\neg t_1\wedge t_2\wedge\neg t_3} + C_{\neg t_1\wedge\neg t_2\wedge t_3} + C_{t_1\wedge\neg t_2\wedge t_3} + C_{\neg t_1\wedge t_2\wedge t_3} + C_{t_1\wedge t_2\wedge t_3}$	$C_{t_1} + C_{t_2} + C_{t_3} - 2C_{t_1\wedge t_2} - C_{t_1\wedge t_3} - C_{t_2\wedge t_3} + 2C_{t_1\wedge t_2\wedge t_3}$
30	6	$\neg t_1 \wedge t_2 \wedge t_3 \vee t_1 \wedge (\neg t_2 \vee \neg t_3)$	3	4	3	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{t_1\wedge t_2\wedge\neg t_3} + C_{t_1\wedge\neg t_2\wedge t_3} + C_{\neg t_1\wedge t_2\wedge t_3}$	$C_{t_1} + C_{t_2\wedge t_3} - 2C_{t_1\wedge t_2\wedge t_3}$
31	6	$t_1 \wedge \neg t_2 \wedge \neg t_3 \vee \neg t_1 \wedge t_2 \wedge t_3$	3	2	4	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{\neg t_1\wedge t_2\wedge t_3}$	$C_{t_1} - C_{t_1\wedge t_2} - C_{t_1\wedge t_3} + C_{t_2\wedge t_3}$
32	6	$t_1 \wedge \neg t_2 \wedge \neg t_3 \vee t_2 \wedge (\neg t_1 \vee t_3)$	6	4	5	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{\neg t_1\wedge t_2\wedge\neg t_3} + C_{\neg t_1\wedge t_2\wedge t_3} + C_{t_1\wedge t_2\wedge t_3}$	$C_{t_1} + C_{t_2} - 2C_{t_1\wedge t_2} - C_{t_1\wedge t_3} + 2C_{t_1\wedge t_2\wedge t_3}$
33	6	$t_1 \wedge \neg t_2 \vee t_2 \wedge \neg t_3 \vee \neg t_1 \wedge t_3$	1	6	6	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{\neg t_1\wedge t_2\wedge\neg t_3} + C_{\neg t_1\wedge\neg t_2\wedge t_3} + C_{t_1\wedge t_2\wedge\neg t_3} + C_{t_1\wedge\neg t_2\wedge t_3} + C_{\neg t_1\wedge t_2\wedge t_3}$	$C_{t_1} + C_{t_2} + C_{t_3} - C_{t_1\wedge t_2} - C_{t_1\wedge t_3} - C_{t_2\wedge t_3}$
34	6	$\neg t_1 \wedge (t_2 \vee t_3) \vee t_1 \wedge \neg t_2 \wedge \neg t_3$	3	4	7	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{\neg t_1\wedge t_2\wedge\neg t_3} + C_{\neg t_1\wedge\neg t_2\wedge t_3} + C_{\neg t_1\wedge t_2\wedge t_3}$	$C_{t_1} + C_{t_2} + C_{t_3} - 2C_{t_1\wedge t_2} - 2C_{t_1\wedge t_3} - C_{t_2\wedge t_3} + 2C_{t_1\wedge t_2\wedge t_3}$
35	8	$t_1 \wedge (t_2 \wedge \neg t_3 \vee \neg t_2 \wedge t_3) \vee \neg t_1 \wedge t_2 \wedge t_3$	1	3	4	$C_{t_1\wedge t_2\wedge\neg t_3} + C_{t_1\wedge\neg t_2\wedge t_3} + C_{\neg t_1\wedge t_2\wedge t_3}$	$C_{t_1\wedge t_2} + C_{t_1\wedge t_3} + C_{t_2\wedge t_3} - 3C_{t_1\wedge t_2\wedge t_3}$
36	8	$t_1 \wedge (t_2 \wedge t_3 \vee \neg t_2 \wedge \neg t_3) \vee \neg t_1 \wedge t_2 \wedge \neg t_3$	3	3	6	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{\neg t_1\wedge t_2\wedge\neg t_3} + C_{t_1\wedge t_2\wedge t_3}$	$C_{t_1} + C_{t_2} - 2C_{t_1\wedge t_2} - C_{t_1\wedge t_3} - C_{t_2\wedge t_3} + 3C_{t_1\wedge t_2\wedge t_3}$
37	8	$\neg t_3 \wedge (t_1 \wedge \neg t_2 \vee \neg t_1 \wedge t_2) \vee \neg t_1 \wedge \neg t_2 \wedge t_3$	1	3	7	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{\neg t_1\wedge t_2\wedge\neg t_3} + C_{\neg t_1\wedge\neg t_2\wedge t_3}$	$C_{t_1} + C_{t_2} + C_{t_3} - 2C_{t_1\wedge t_2} - 2C_{t_1\wedge t_3} - 2C_{t_2\wedge t_3} + 3C_{t_1\wedge t_2\wedge t_3}$
38	8	$t_1 \wedge \neg t_2 \wedge \neg t_3 \vee \neg t_1 \wedge (t_2 \vee t_3) \vee t_2 \wedge t_3$	3	5	7	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{\neg t_1\wedge t_2\wedge\neg t_3} + C_{\neg t_1\wedge\neg t_2\wedge t_3} + C_{\neg t_1\wedge t_2\wedge t_3} + C_{t_1\wedge t_2\wedge t_3}$	$C_{t_1} + C_{t_2} + C_{t_3} - 2C_{t_1\wedge t_2} - 2C_{t_1\wedge t_3} - C_{t_2\wedge t_3} + 3C_{t_1\wedge t_2\wedge t_3}$
39	10	$t_1 \wedge (t_2 \wedge t_3 \vee \neg t_2 \wedge \neg t_3) \vee \neg t_1 \wedge (t_2 \wedge \neg t_3 \vee \neg t_2 \wedge t_3)$	1	4	7	$C_{t_1\wedge\neg t_2\wedge\neg t_3} + C_{\neg t_1\wedge t_2\wedge\neg t_3} + C_{\neg t_1\wedge\neg t_2\wedge t_3} + C_{t_1\wedge t_2\wedge t_3}$	$C_{t_1} + C_{t_2} + C_{t_3} - 2C_{t_1\wedge t_2} - 2C_{t_1\wedge t_3} - 2C_{t_2\wedge t_3} + 4C_{t_1\wedge t_2\wedge t_3}$

References

[Abbassi et al., 2013] Abbassi, Z., Mirrokni, V. S., and Thakur, M. (2013). Diversity maximization under matroid constraints. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 32–40, New York, NY, USA. ACM.

[Aitchison and Shen, 1980] Aitchison, J. and Shen, S. M. (1980). Logistic-normal distributions: some properties and uses. *Biom.*, 67(2):261–272.

[Ames and Naaman, 2007] Ames, M. and Naaman, M. (2007). Why we tag: Motivations for annotation in mobile and online media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 971–980, New York, NY, USA. ACM.

[Antwi and Viktor, 2014] Antwi, D. K. and Viktor, H. L. (2014). Building smart cubes for reliable and faster access to data. In *Data Warehousing and Knowledge Discovery - 16th International Conference, DaWaK 2014, Munich, Germany, September 2-4, 2014. Proceedings*, pages 254–265.

[Apté et al., 1994] Apté, C., Damerau, F., and Weiss, S. M. (1994). Automated learning of decision rules for text categorization. *ACM Trans. Inf. Syst.*, 12(3):233–251.

[Baeza-Yates and Ribeiro-Neto, 1999] Baeza-Yates, R. A. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Bansal and Koudas, 2007] Bansal, N. and Koudas, N. (2007). Searching the blogosphere. In *Tenth International Workshop on the Web and Databases, WebDB 2007, Beijing, China, June 15, 2007*.

[Ben-Yitzhak et al., 2008] Ben-Yitzhak, O., Golbandi, N., Har'El, N., Lempel, R., Neumann, A., Ofek-Koifman, S., Sheinwald, D., Shekita, E. J., Sznajder, B., and Yogev, S. (2008). Beyond basic faceted search. In *Proceedings of the International Conference on Web Search and Web Data Mining, WSDM 2008, Palo Alto, California, USA, February 11-12, 2008*, pages 33–44.

[Bi and Cho, 2013] Bi, B. and Cho, J. (2013). Automatically generating descriptions for resources by tag modeling. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, CIKM '13, pages 2387–2392, New York, NY, USA. ACM.

[Blei and Lafferty, 2005] Blei, D. M. and Lafferty, J. D. (2005). Correlated topic models. In *Advances in Neural Information Processing Systems 18*, NIPS '05.

[Blei and Lafferty, 2007] Blei, D. M. and Lafferty, J. D. (2007). A correlated topic model of science. *Ann. Appl. Stat.*, 1:17–35.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

[Brooks and Montanez, 2006] Brooks, C. H. and Montanez, N. (2006). Improved annotation of the blogosphere via autotagging and hierarchical clustering. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 625–632, New York, NY, USA. ACM.

[Budzik and Hammond, 1999] Budzik, J. and Hammond, K. (1999). Watson: Anticipating and contextualizing information needs. In *62nd annual meeting of the American society for information science*, pages 727–740.

[Caballero et al., 2012] Caballero, K. L., Barajas, J., and Akella, R. (2012). The generalized dirichlet distribution in enhanced topic detection. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 773–782, New York, NY, USA. ACM.

[Candan and Sapino, 2010] Candan, K. S. and Sapino, M. L. (2010). *Data Management for Multimedia Retrieval*. Cambridge University Press, New York, NY, USA.

[Carbonell and Goldstein, 1998] Carbonell, J. G. and Goldstein, J. (1998). The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on*

Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia, pages 335–336.

[Cattuto et al., 2009] Cattuto, C., Barrat, A., Baldassarri, A., Schehr, G., and Loreto, V. (2009). Collective dynamics of social annotation. *Proc. Natl. Acad. Sci.*, 106(26):10511–10515.

[Côté et al., 2010] Côté, R., Reisinger, F., Martens, L., Barsnes, H., Vizcaino, J. A., and Hermjakob, H. (2010). The Ontology Lookup Service: bigger and better. *Nucleic Acids Res*, 38:W155–W160.

[Cutting et al., 1993] Cutting, D. R., Karger, D. R., and Pedersen, J. O. (1993). Constant interaction-time Scatter/Gather browsing of very large document collections. In *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Pittsburgh, PA, USA, June 27 - July 1, 1993*, pages 126–134.

[Cutting et al., 1992] Cutting, D. R., Pedersen, J. O., Karger, D. R., and Tukey, J. W. (1992). 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. Copenhagen, Denmark, June 21-24, 1992*, pages 318–329.

[Das and Martins, 2007] Das, D. and Martins, A. F. T. (2007). A survey on automatic text summarization. Technical report, Literature Survey for the Language and Statistics II course at Carnegie Mellon University.

[Dash et al., 2008] Dash, D., Rao, J., Megiddo, N., Ailamaki, A., and Lohman, G. M. (2008). Dynamic faceted search for discovery-driven analysis. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*, pages 3–12.

[Deolalikar, 2014] Deolalikar, V. (2014). Distance or coverage?: Retrieving knowledge-rich documents from enterprise text collections. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 1771–1774.

[Ebbert et al., 2003] Ebbert, J. O., Dupras, D. M., and Erwin, P. J. (2003). Searching the medical literature using PubMed: a tutorial. *Mayo Clinic proceedings. Mayo Clinic*, 78(1):87–91.

[Efthimiadis, 1995] Efthimiadis, E. N. (1995). User choices: A new yardstick for the evaluation of ranking algorithms for interactive query expansion. *Inf. Process. Manage.*, 31(4):605–620.

[Fagan, 2013] Fagan, J. C. (2013). Usability studies of faceted browsing: a literature review. *Inf. Technol. Libr.*, 29(2):58–66.

[Garnes, 2009] Garnes, Ø. L. (2009). Feature selection for text categorisation. Master's thesis, Norwegian University of Science and Technology, Department of Computer and Information Science.

[Gelbukh et al., 2003] Gelbukh, A. F., Alexandrov, M., Bourek, A., and Makagonov, P. (2003). Selection of representative documents for clusters in a document collection. In *Natural Language Processing and Information Systems, 8th International Conference on Applications of Natural Language to Information Systems, June 2003, Burg (Spreewald), Germany*, pages 120–126.

[Goodwin et al., 2012] Goodwin, J. C., Cohen, T., and Rindflesch, T. C. (2012). Discovery by scent: Discovery browsing system based on the Information Foraging Theory. In *2012 IEEE International Conference on Bioinformatics and Biomedicine Workshops, BIBMW 2012, Philadelphia, USA, October 4-7, 2012*, pages 232–239.

[Goorha and Ungar, 2010] Goorha, S. and Ungar, L. (2010). Discovery of significant emerging trends. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 57–64, New York, NY, USA. ACM.

[Gray et al., 1997] Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., and Pirahesh, H. (1997). Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Min. Knowl. Discov.*, 1(1):29–53.

[Gupta and Lehal, 2010] Gupta, V. and Lehal, G. S. (2010). A survey of text summarization extractive techniques. *Journal of Emerging Technologies in Web Intelligence*, 2(3):258–268.

[Harinarayan et al., 1996] Harinarayan, V., Rajaraman, A., and Ullman, J. D. (1996). Implementing data cubes efficiently. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, SIGMOD '96, pages 205–216, New York, NY, USA. ACM.

[Hearst, 2006] Hearst, M. A. (2006). Clustering versus faceted categories for information exploration. *Commun. ACM*, 49(4):59–61.

[Hearst, 2009] Hearst, M. A. (2009). *Search User Interfaces.* Cambridge University Press, New York, NY, USA, 1st edition.

[Inokuchi and Takeda, 2007] Inokuchi, A. and Takeda, K. (2007). A method for online analytical processing of text data. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007*, pages 455–464.

[Jansen et al., 2000] Jansen, B. J., Spink, A., and Saracevic, T. (2000). Real life, real users, and real needs: a study and analysis of user queries on the web. *Inf. Process. Manag.*, 36(2):207–227.

[Jin et al., 2010] Jin, X., Han, J., Cao, L., Luo, J., Ding, B., and Lin, C. X. (2010). Visual cube and on-line analytical processing of images. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 849–858, New York, NY, USA. ACM.

[Kashyap et al., 2010] Kashyap, A., Hristidis, V., and Petropoulos, M. (2010). Facetor: cost-driven exploration of faceted query results. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, pages 719–728.

[Ke et al., 2009] Ke, W., Sugimoto, C. R., and Mostafa, J. (2009). Dynamicity vs. effectiveness: studying online clustering for Scatter/Gather. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009*, pages 19–26.

[Kim and Rieh, 2011] Kim, Y.-M. and Rieh, S. Y. (2011). User perceptions of the role and value of tags. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 671–674, New York, NY, USA. ACM.

[Kipp and Campbell, 2010] Kipp, M. E. and Campbell, D. G. (2010). Searching with tags: Do tags help users find things? *Knowl. Organ.*, 37(4):239–255.

[Krestel et al., 2009] Krestel, R., Fankhauser, P., and Nejdl, W. (2009). Latent dirichlet allocation for tag recommendation. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 61–68, New York, NY, USA. ACM.

[Lakshmanan et al., 2002] Lakshmanan, L. V. S., Pei, J., and Han, J. (2002). Quotient cube: how to summarize the semantics of a data cube. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pages 778–789. VLDB Endowment.

[Lee, 1999] Lee, L. (1999). Measures of distributional similarity. In *27th Annual Meeting of the Association for Computational Linguistics, Univeristy of Maryland, College Park, Maryland, USA, 20-26 June 1999.*

[Lemire et al., 2010] Lemire, D., Kaser, O., and Aouiche, K. (2010). Sorting improves word-aligned bitmap indexes. *Data Knowl. Eng.*, 69(1):3–28.

[Li et al., 2010] Li, C., Yan, N., Roy, S. B., Lisham, L., and Das, G. (2010). Facetedpedia: dynamic generation of query-dependent faceted interfaces for wikipedia. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 651–660.

[Li et al., 2007] Li, W., Blei, D. M., and McCallum, A. (2007). Nonparametric bayes pachinko allocation. In *UAI 2007, Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, Vancouver, BC, Canada, July 19-22, 2007*, pages 243–250.

[Li and McCallum, 2006] Li, W. and McCallum, A. (2006). Pachinko allocation: Dag-structured mixture models of topic correlations. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, pages 577–584.

[Li et al., 2004] Li, X., Han, J., and Gonzalez, H. (2004). High-dimensional OLAP: a minimal cubing approach. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 528–539. VLDB Endowment.

[Lin et al., 2008] Lin, C. X., Ding, B., Han, J., Zhu, F., and Zhao, B. (2008). Text cube: Computing IR measures for multidimensional text database analysis. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 905–910.

[Lu et al., 2009] Lu, Z., Kim, W., and Wilbur, W. (2009). Evaluation of query expansion using mesh in pubmed. *Inform. Retrieval*, 12(1):69–80.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA.

[Mendoza et al., 2015] Mendoza, M., Alegra, E., Maca, M., Cobos, C., and Len, E. (2015). Multidimensional analysis model for a document warehouse that includes textual measures. *Decis. Support Syst.*, 72:44 – 59.

[Millen et al., 2006] Millen, D. R., Feinberg, J., and Kerr, B. (2006). Dogear: Social bookmarking in the enterprise. In *Proceedings of the 2006 Conference on Human Factors in Computing Systems, CHI 2006, Montréal, Québec, Canada, April 22-27, 2006*, pages 111–120. ACM.

[Mosa and Yoo, 2013] Mosa, A. and Yoo, I. (2013). A study on pubmed search tag usage pattern: Association rule mining of a full-day pubmed query log. *BMC Med. Inform. Decis. Mak.*, 13(1).

[Mu et al., 2014] Mu, X., Lu, K., and Ryu, H. (2014). Explicitly integrating mesh thesaurus help into health information retrieval systems: An empirical user study. *Inf. Process. Manag.*, 50(1):24–40.

[Paisley et al., 2012] Paisley, J., Wang, C., and Blei, D. M. (2012). The discrete infinite logistic normal distribution. *Bayesian Analysis*, 7(4):997–1034.

[Pirolli et al., 2000] Pirolli, P., Card, S. K., and Van Der Wege, M. M. (2000). The effect of information scent on searching information: Visualizations of large tree structures. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '00, pages 161–172, New York, NY, USA. ACM.

[Popescul and Ungar, 2000] Popescul, A. and Ungar, L. H. (2000). Automatic labeling of document clusters.

[Pratt et al., 1999] Pratt, W., Hearst, M. A., and Fagan, L. M. (1999). A knowledge-based approach to organizing retrieved documents. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence, July 18-22, 1999, Orlando, Florida, USA.*, pages 80–85.

[Radev et al., 2004] Radev, D. R., Jing, H., Styś, M., and Tam, D. (2004). Centroid-based summarization of multiple documents. *Inf. Process. Manage.*, 40(6):919–938.

[Rajaraman and Ullman, 2011] Rajaraman, A. and Ullman, J. D. (2011). *Mining of Massive Datasets.* Cambridge University Press, New York, NY, USA.

[Ramdeen and Hemminger, 2012] Ramdeen, S. and Hemminger, B. M. (2012). A tale of two interfaces: How facets affect the library catalog search. *J. Am. Soc. Inf. Sci. Technol.*, 63(4):702–715.

[Ravat et al., 2008] Ravat, F., Teste, O., Tournier, R., and Zurfluh, G. (2008). Top keyword: An aggregation function for textual document olap. In *Proceedings of the 10th International Conference on Data Warehousing and Knowledge Discovery*, DaWaK '08, pages 55–64, Berlin, Heidelberg. Springer-Verlag.

[Raymond, 1984] Raymond, D. R. (1984). Personal data structuring in videotex. Technical Report CS-84-7, Cheriton Sch. Comp. Sci., Univ. Waterloo.

[Roy et al., 2008] Roy, S. B., Wang, H., Das, G., Nambiar, U., and Mohania, M. K. (2008). Minimum-effort driven dynamic faceted search in structured databases. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*, pages 13–22.

[Sandhaus, 2008] Sandhaus, E. (2008). The New York Times Annotated Corpus.

[Santos et al., 2010] Santos, R. L. T., Macdonald, C., and Ounis, I. (2010). Exploiting query reformulations for web search result diversification. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 881–890.

[Santos et al., 2015] Santos, R. L. T., Macdonald, C., and Ounis, I. (2015). Search result diversification. *Found. Trends Inf. Retr.*, 9(1):1–90.

[Sebastiani, 2002] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47.

[Sedgewick and Wayne, 2011] Sedgewick, R. and Wayne, K. (2011). *Algorithms, 4th Edition.*

[Shiri et al., 2011] Shiri, A., Ruecker, S., Doll, L., Bouchard, M., and Fiorentino, C. (2011). An evaluation of thesaurus-enhanced visual interfaces for multilingual digital libraries. In Gradmann, S., Borri, F., Meghini, C., and Schuldt, H., editors, *Res. Adv. Technol. Digit. Libr.*, volume 6966 of *Lect. Notes Comp. Sci.*, pages 236–243. Springer.

[Simitsis et al., 2008] Simitsis, A., Baid, A., Sismanis, Y., and Reinwald, B. (2008). Multidimensional content exploration. *PVLDB*, 1(1):660–671.

[Sismanis et al., 2002] Sismanis, Y., Deligiannakis, A., Roussopoulos, N., and Kotidis, Y. (2002). Dwarf: shrinking the petacube. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, SIGMOD '02, pages 464–475, New York, NY, USA. ACM.

[Teevan et al., 2008] Teevan, J., Dumais, S. T., and Gutt, Z. (2008). Challenges for supporting faceted search in large, heterogeneous corpora like the web. In *Proceedings of the 2nd Workshop on Human-Computer Interaction and Information Retrieval*, HCIR '08, pages 67–69.

[Tunkelang, 2009] Tunkelang, D. (2009). *Faceted Search*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers.

[Vee et al., 2008] Vee, E., Srivastava, U., Shanmugasundaram, J., Bhat, P., and Yahia, S. (2008). Efficient computation of diverse query results. In *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, México*, pages 228–236.

[Wang et al., 2002] Wang, W., Lu, H., Feng, J., and Yu, J. X. (2002). Condensed cube: An efficient approach to reducing data cube size. In *Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, February 26 - March 1, 2002*, ICDE '02, pages 155–165.

[Wei et al., 2013] Wei, B., Liu, J., Zheng, Q., Zhang, W., Fu, X., and Feng, B. (2013). A survey of faceted search. *J. Web Eng.*, 12(1&2):41–64.

[White, 2001] White, J. (2001). ACM opens portal to computing literature. *Commun. ACM*, 44(7):14–16, 28.

[Witten et al., 1999] Witten, I. H., Paynter, G. W., Frank, E., Gutwin, C., and Nevill-Manning, C. G. (1999). Kea: Practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM Conference on Digital Libraries*, DL '99, pages 254–255, New York, NY, USA. ACM.

[Yang and Pedersen, 1997] Yang, Y. and Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Yee et al., 2003] Yee, K., Swearingen, K., Li, K., and Hearst, M. A. (2003). Faceted metadata for image search and browsing. In *Proceedings of the 2003 Conference on Human Factors in Computing Systems, CHI 2003, Ft. Lauderdale, Florida, USA, April 5-10, 2003*, pages 401–408.

[Zamir and Etzioni, 1999] Zamir, O. and Etzioni, O. (1999). Grouper: A dynamic clustering interface to web search results. In *Proceedings of the Eighth International Conference* *on World Wide Web*, WWW '99, pages 1361–1374, New York, NY, USA. Elsevier North-Holland, Inc.

[Zhai et al., 2003] Zhai, C., Cohen, W. W., and Lafferty, J. D. (2003). Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 28 - August 1, 2003, Toronto, Canada*, pages 10–17.

[Zhang et al., 2011] Zhang, D., Zhai, C., and Han, J. (2011). Mitexcube: Microtextcluster cube for online analysis of text cells. In *Proceedings of the 2011 Conference on Intelligent Data Understanding, CIDU 2011, October 19-21, 2011, Mountain View, California, USA*, pages 204–218.

[Zhang et al., 2009] Zhang, D., Zhai, C., Han, J., Srivastava, A., and Oza, N. (2009). Topic modeling for olap on multidimensional text databases: Topic cube and its applications. *Stat. Anal. Data Min.*, 2(56):378–395.

[Zhang and Marchionini, 2005] Zhang, J. and Marchionini, G. (2005). Evaluation and evolution of a browse and search interface: relation browser. In *Proceedings of the 2005 National Conference on Digital Government Research, DG.O 2005, Atlanta, Georgia, USA, May 15-18, 2005*, pages 179–188.