

# Exploiting Asymmetry in Hierarchical Topic Extraction

Sreenivas Gollapudi  
Ebrary, Inc.

Sreenivas.Gollapudi@ebrary.com

Rina Panigraphy  
Stanford University

rinap@cs.stanford.edu

## ABSTRACT

Topic or feature extraction is often used as an important step in document classification and text mining. Topics are succinct representation of content in a document collection and hence are very effective when used as content identifiers in peer-to-peer systems and other large scale distributed content management systems. Effective topic extraction is dependent on the accuracy of term clustering that often has to deal with problems like *synonymy* and *polysemy*. Retrieval techniques based on spectral analysis like Latent Semantic Indexing (LSI) are often used to effectively solve these problems. Most of the spectral retrieval schemes produce term similarity measures that are symmetric and often, not an accurate characterization of term relationships. Another drawback of LSI is its running time that is polynomial in the dimensions of the  $m \times n$  matrix,  $A$ . This can get prohibitively large for some IR applications. In this paper, we present efficient algorithms using the technique of Locality-Sensitive Hashing (LSH) to extract topics from a document collection based on the asymmetric relationships between terms in a collection. The relationship is characterized by the term co-occurrences and other higher-order similarity measures. Our LSH based scheme can be viewed as a simple alternative to LSI. We show the efficacy of our algorithms via experiments on a set of large documents. An interesting feature of our algorithms is that it produces a natural hierarchical decomposition of the topic space instead of a flat clustering.

## Categories and Subject Descriptors

H.3.3 [Information Search and Retrieval]: Clustering, Hashing

## General Terms

Algorithms, Experimentation

## Keywords

Latent Semantic Indexing, Locality-Sensitive Hashing

## 1. INTRODUCTION

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM '06, November 6–11, Arlington, Virginia USA

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

Topics are succinct content identifiers that can be effectively used in various applications such as document classification and clustering. One of the desired characteristics of a term clustering scheme is the ability to group terms based on their semantic relationships. To this end, LSI uses singular value decomposition (SVD) to “cluster” terms and documents based on their semantic content. An excellent exposition of LSI and its applications in Information Retrieval can be found in [17] and references therein. The key feature of LSI is to map the documents from the  $m$ -dimensional term space to (often significantly) lower  $k$ -dimensional topic space. Each topic can be specified by a weighted set of terms and typically, corresponds to a left eigenvector of the original term-document matrix,  $A$ . When the term-document matrix is large, the exact computation of the SVD of the matrix and hence the decomposition of the original term space into topic space becomes prohibitively expensive ( $\min\{mn^2, m^2n, mnk\}$ ). To overcome this problem, most variants of LSI [6, 8] deal with the best rank  $k$  approximation  $D^*$  of  $A$  such that

$$\|A - D^*\|_F = \min_{\text{rank}(D)=k} \|A - D\|_F,$$

where  $\|\cdot\|_F$  is the Frobenius norm of the matrix. The reader is referred to the theorem of Eckert and Young [1] that shows the existence of such a minimum. The r.h.s of the above equation is the sum of the smallest  $n - (k + 1)$  singular values of  $A$ .

In this paper, we focus on using relationships between terms to compute the topic subspace. We consider similar words to be related to similar terms. An interesting feature of our algorithms is that it produces a natural hierarchical decomposition of the topic space instead of flat clustering. We compute the topic set by using *Locality-Sensitive Hashing (LSH)* [11, 14] to compute the similarity between terms rather than measures like cosine similarity commonly used in vector space models. A locality-sensitive hashing scheme is a distribution on a family  $\mathcal{H}$  of hash functions when applied to a collection  $\mathcal{C}$  of entities yields

$$\Pr_{h \in \mathcal{H}} [h(x) = h(y)] = \text{sim}(x, y),$$

where  $\text{sim}(x, y) \in [0, 1]$  is some *similarity function* defined on the collection  $\mathcal{C}$  [4]. The primary advantage of this approach is its efficiency. The pairwise similarity between  $m$  terms can be computed in time proportional to the number of non-zero entries in the matrix,  $O(|M_{mn}|_0)$  versus the  $O(m^2n)$  it takes for computing the cosine similarity in vector space models. We will describe LSH in more detail in section 3. LSH lacks the advantage of SVD to compute similarity of two objects by not only comparing the objects themselves, but also recursing to include the similarity of sets of ob-

jects similar to these objects. There is work to show that LSI exploits higher-order term co-occurrences [18]. We overcome the drawback by computing *higher-order similarities* between terms. Higher-order term similarity essentially defines similarity between two terms as a function of the number of terms that are similar to both terms. This procedure is detailed in Section 4.1.

In this work, we adopt an *asymmetric* measure for similarity which, in our observations, is more appropriate for document collections. In information theoretic terms, a pairwise symmetric measure of term similarity which implies the *mutual information* contained in both terms is roughly the same, i.e.,  $A$  tells as much about  $B$  as much as  $B$  tells about  $A$ . In reality, this is not always true. Term relationships are usually characterized by generalization and specialization of a concept. For any term pair  $(t_a, t_b)$ ,  $t_a < t_b$  if  $t_a$  is a *hypernym* of  $t_b$ ;  $t_a > t_b$  if  $t_a$  is a *hyponym* of  $t_b$ . For example, for the term pair  $(network, protocol)$ , every occurrence of the term *protocol* might co-exist with the term *network*, but the inverse might not be true. The term *network* might occur along with other terms such as *computer*. We use a similarity function that supports the asymmetric relationship between terms in the collection. One such function is

$$sim_A(A, B) = \Pr(B|A) = \frac{|A \cap B|}{|A|}.$$

This similarity measure can be interpreted as the natural conditional probability of term  $B$  occurring in a random document given that it contains  $A$ . This measure of correlating two sets has been used widely in several areas including data mining for mining association rules between database columns. We believe the methods we proposed for computing term similarity using this measure is applicable to not only information retrieval, but also to other contexts such as recommendation systems and data mining. Such a metric rightfully allows for topics to be hierarchically related instead of a flat structure. Our algorithms naturally produce such a hierarchy.

## 1.1 Contributions of this Study

In this study, we propose algorithms for hierarchical topic extraction from a collection of documents using the asymmetric relationships between terms. The algorithms are efficient compared to schemes that are used to compute term similarities. The efficiencies are obtained by sampling only a small subset of the original term-document matrix and then computing the similarity between two terms using LSH instead of cosine similarities. We propose novel algorithms for bag similarities where the running time depends only on the number of distinct entries in the multi-set. Furthermore, we extend the term similarity computation to include higher-order similarity measures to approximate computationally intensive schemes like LSI. The running time of our algorithms is proportional to the number of non-zero entries in the truncated term-document space. The details of this scheme are presented in Sections 3 and 4. Finally, in Section 6, we show the efficacy of our algorithms via experiments on large document sets.

## 1.2 Related Work

Some of the work on topic extraction has broadly focused on concept learning [22, 9] while some others studied semantic similarity between terms (see [3] and references within).

Besides, there is a plethora of work in this area based on symmetric similarity measures.

The most relevant work on LSH for document clustering is that of Haveliwala, Gionis, and Indyk [12]. In that work, they propose an algorithm to cluster URLs based on LSH. Our algorithm differs from theirs in several respects: first, we use asymmetric similarity to compare terms; second, since our LSH is performed on bags with large counts, their algorithms become impractical. Their work looks at every element in the multi-set to compute bag similarity. This can get prohibitively large; third, their work focuses on document clustering and does not address topic extraction.

Drineas and others [8] propose an approximation algorithm to efficiently compute the SVD on a random submatrix of a given matrix. However, it is only applicable to symmetric matrices.

Jeh and Widom [15] propose a fixed-point scheme to compute similarity between objects based on concepts that are similar to ours. Again, their analysis relies on symmetric relationship between objects and consider all term-pairs in their computations. In this study, we circumvent similarity computation between all term pairs. Dhillon, Mallela and Kumar [7] provide a feature clustering algorithm based on information-theoretic analysis while Kontostathis and Potenger [18] show that higher-order co-occurrences, used by LSI and our algorithms, are important for term clustering.

## 2. MODELS AND PRELIMINARIES

In this section, we will present the model and definitions that form the basis of our algorithms to extract topics and from the document classification. We model the asymmetric relationship between terms by a weighted directed graph  $G = (V, E)$  with each term representing a vertex in  $G$  and an edge between vertex  $u$  and vertex  $v$  exists if terms corresponding to vertices  $u$  and  $v$  co-occur in the collection. Note that the weight on the edge  $(u, v)$  is not necessarily equal to the weight on the edge  $(v, u)$ . Clustering nodes in such graphs can be very hard since the graphs can be very large. We propose randomized algorithms to group nodes that are similar to each other.

### 2.1 Overview of our algorithmic approach

We now present an overview of our algorithms, which we will detail in the following sections. Before we start processing a term-document matrix, we rank the terms in the collection based on their average TFIDF scores and consider the top  $k$  terms in the analysis. Typically, it is the TFIDF score of the term in the document of interest  $\log(1 + t f_{ij}) \frac{\log(1+N)}{|D_i|}$ .

The algorithm to compute the topics based on higher-order term similarities runs in two phases. In the first phase, we compute the term-term similarity matrix as follows:

1. Each term is represented as a document vector where element  $i$  in the vector is the weight of the term in document  $i$ .
2. Next, we compute the sketch for each term as a concatenation of  $l$  LSH hashes on the document vector.
3. We compute similarity between two given terms  $sim(t_1, t_2)$  as the number of positions in which their LSH values match. We will show that this can step can be efficiently implemented without looking at all term pairs.

In the second phase of the algorithms, we consider the similarity between sets of terms that are similar to  $t_1$  and  $t_2$  to compute the higher-order similarity.

1. Each term is represented by a term vector whose length  $k \ll m$ . An element  $i$  in this vector is a measure of how similar the given term is to term  $i$ . Note that like in the previous phase, this vector is not a unit vector.
2. Next we compute sketch of each term by concatenation of  $l$  LSH hashes on the term vector and the similarity is computed in the same way as in the first phase.

Based on the term-term similarity matrix, we generate the directed graph and compute strongly connected components (SCCs) in this graph and the resulting kernel-DAG (directed acyclic graph) with the strongly connected components as nodes. Each SCC is declared as a topic with the relationship between two topics encapsulated by the direction of the edge between edges in the kernel-DAG. A topic  $a$  that generalizes another topic  $b$  is represented by the edge  $(a, b)$  in the kernel-DAG.

The details of the two phases are presented in Sections 3 and 4 respectively. We start with an exposition of the theory behind locality-sensitive hashing and show how it can be used to design fast algorithms for similarity estimation.

### 3. LOCALITY-SENSITIVE HASH FUNCTIONS

*Definition 1.* A locality-sensitive hashing scheme [4] is a distribution on a family  $\mathcal{H}$  of hash functions operating on a collection of objects, such that for any two objects  $x, y$ ,

$$\Pr_{h \in \mathcal{H}} [h(x) = h(y)] = \text{sim}(x, y).$$

Here  $\text{sim}(x, y)$  is some similarity function between objects in the collection.

A commonly used similarity measure for comparing sets  $A$  and  $B$  is

$$\text{sim}(A, B) = \frac{|A \cap B|}{|A \cup B|}. \quad (1)$$

There is a well-known locality-sensitive hashing function – Min-wise independent permutation [2] – that can be used for this similarity measure. Clearly,  $\text{sim}(A, B)$  is symmetric. For clarity, we will denote this measure of similarity by  $\text{sim}_S(A, B)$ . Given a set  $A$  of elements from a universe  $U$ , its min-wise independent permutation denoted  $MH(A)$  is computed as follows: let  $R(x), x \in U$  be a real valued (hash) function that maps elements from the universe  $U$  to a real number randomly and uniformly in the interval  $[0, 1]$ .

*Definition 2.* A Min-wise independent permutation is defined by  $MH(A) = \text{argmin}_x \{R(x) | x \in A\}$ . Essentially,  $MH(A)$  is the element in  $A$  whose hash value into the interval  $[0, 1]$  is minimum.

Observe that the hash function  $R$  used to map elements to real numbers must be performed *consistently*; that is,  $R$  applied on the same element must result in the same value each time regardless of which set(s) it appears in.

Alternately, instead of using the function that maps each element to a real number, we can use a random permutation of all the elements in  $U$ , and define  $MH(A)$  to be the leftmost element of  $A$  in this permutation. As mentioned earlier, it is well known that

**THEOREM 1.** [2] For any two sets  $A$  and  $B$ ,  $\Pr[MH(A) = MH(B)] = \frac{|A \cap B|}{|A \cup B|}$

A simple extension of this similarity measure and the hash function from sets to bags has been shown in [12]. Given a bag (multiset), create a new set with distinct elements for each copy of a given element in the bag. Essentially, if  $f_x$  is the number of occurrences of the element  $x$  in a bag, replace  $x$  by the pairs  $\{(x, i) | 1 \leq i \leq f_x\}$ . It is easy to see that the size of the intersection and union of the sets thus obtained are same as the bag intersection and bag union. This gives a simple generalization of min wise independent permutation that works for bags. However, unfortunately the time complexity of computing such a hash function grows linearly in the total number of occurrences of the different elements. So, if the number of occurrences of a certain element is a million, then a million operations have to be performed for that element. Also, this method does not work if the number of occurrences (or weight) of an element is allowed to be fractional. One method to deal with non-integral weights is to multiply these weights by a large number and round them to the nearest integer. However, as pointed before, multiplying by large number makes the number of occurrences large thus resulting in a large running time.

We present two methods to provide a fast running time even if the number of occurrences of the different elements is large. Unlike the earlier algorithm, our running time is strictly polynomial in the size of the bit representations of the frequencies. The second method also works for weighted sets with fractional weights.

#### 3.1 Method 1

Since the bag elements have been ordered to pairs of the form  $(x, i)$ , we will work with a real valued hash function  $R$  that maps such pairs to a random value in  $[0, 1]$ .

Given a bag  $A$ , where an element  $x$  has a frequency  $f_x$ , we need to compute

$$MH(A) = \text{argmin}_x \{R(x, i) | x \in A, 1 \leq i \leq f_x\}.$$

This computation can be viewed as first computing,

$$MH_e(x, f_x) = \text{argmin}_i \{R(x, i) | 1 \leq i \leq f_x\},$$

where  $MH_e(x, f_x)$  denotes the min-hash value of the  $f_x$  copies of  $x$ . Let  $y = \text{argmin}_x \{MH_e(x, f_x) | x \in A\}$ . And then computing,

$$MH(A) = (y, MH_e(y, f_y)).$$

We will show how to compute  $MH_e(x, f_x)$  in time  $\log f_x$  instead of  $f_x$ .

Visualize the process of hashing the different copies of  $x$  onto the real line sequentially. An important observation is that since we want to find the copy of  $x$  that hashes to the smallest real number, we can maintain the smallest hash value seen so far and ignore the computations on subsequent elements that hash to its right, i.e., larger values.

That is, given that  $MH_e(x, j) = r$ , with probability  $1 - r$ , each of  $(x, j + 1), (x, j + 2), \dots$  will hash to a value larger than  $r$  which will leave the  $MH_e$  value unchanged. In fact, we can directly compute the distribution on the number of elements that will hash to the right of (greater than)  $r$  before the next one hashes to its left. This is essentially a random integer with the inverse geometric distribution with parameter  $1 - r$ . Denote this by  $G_{1-r}^{-1}$ . Let  $G_{1-r}^{-1}(x, i)$  denote the

distribution seeded by  $(x, i)$ . This ensures the same value from the distribution for the same seed.

We have the following algorithm to compute  $MH_e(x, f_x)$ :

---

**Algorithm 1** COMPUTEMINHASH,  $MH_e(x, f_x)$

---

```

1:  $i \leftarrow 0, r \leftarrow 1$ 
2: while  $i \leq f_x$  do
3:    $skip \leftarrow G_{1-r}^{-1}(x, i)$  {number of elements that will hash
     to the right of  $r$ }
4:    $mh \leftarrow i$ 
5:    $i \leftarrow i + skip$ 
6:    $r \leftarrow rU_{[0,1]}(x, i)$  {uniform random number in the
     range  $[0, 1]$ , seeded by  $(x, i)$ }
7: end while
8: return  $MH_e(x, f_x) = mh$ 

```

---

Since algorithm 1 produces  $MH_e(x, f_x)$  with the same distribution as before, we have the following theorem.

**THEOREM 2.** For any two bags  $A$  and  $B$ , for the hash functions defined above,  $\Pr[MH(A) = MH(B)] = \frac{|A \cap B|}{|A \cup B|}$

### 3.2 Method 2

Although Method 1 takes only  $\log f_x$  time for each element  $x$ , it still requires an iterative computation that picks random numbers from an inverse geometric distribution.

We present an alternate method that is much simpler but realizes the desired similarity measure with a small error. We will argue that if the bag sizes are large than the similarity measure it realizes is very close to the desired similarity measure. To achieve this we need to make use of an upper bound  $F$  on the frequency of any element in a bag. We then normalize the weight of each bag element to a value between zero and one by setting  $w_x = f_x/F$ .

Let  $\tilde{A}$  and  $\tilde{B}$ , denote the normalized bags where an element has frequency equal to its weight.

$$|\tilde{A}| = \sum w_x(A), |\tilde{B}| = \sum w_x(B)$$

Instead of working with bag  $A$ , we produce a set  $A'$  that includes each element  $x \in A$  with probability  $w_x$ , as follows. Let  $R_2(x)$  being a random (consistent) function that maps elements to a real number in  $[0, 1]$  such that  $A' = \{x \in A \mid R_2(x) \leq w_x\}$ . To compare the two bags  $A$  and  $B$ , we simply compute  $sim_S(A', B')$ .

We summarize the algorithm for computing the min-hash value of a bag.

---

**Algorithm 2** COMPUTEMINHASH,  $MH(A)$

---

```

1: normalize the bag  $A$  to produce  $\tilde{A}$ .
2: produce set  $A' = \{x \in A \mid R_2(x) \leq w_x\}$ .
3: return  $MH(A')$  {Since  $A'$  is a set, use min-wise independent permutations}.

```

---

Continuing the analysis to quantify the error in the similarity computation introduced by this method, it is easy to see that given two bags  $A$  and  $B$ ,

$$E[|A' \cap B'|] = \sum_x \min(w_x(A), w_x(B))$$

$$E[|A' \cup B'|] = \sum_x \max(w_x(A), w_x(B))$$

Unfortunately  $E[\frac{|A' \cap B'|}{|A' \cup B'|}]$  may not be equal to

$$\frac{\sum_x \min(w_x(A), w_x(B))}{\sum_x \max(w_x(A), w_x(B))} = \frac{|A \cap B|}{|A \cup B|}$$

However, it is possible to show that if  $|\tilde{A}|$  or  $|\tilde{B}|$  is large, then the similarity computation holds within a small error.

**THEOREM 3.** For any two bags  $A$  and  $B$ , and  $0 < \epsilon < 1$ , if one of  $|\tilde{A}|$  and  $|\tilde{B}|$  is bounded below by  $M$ , then for the hash functions defined above, with probability at least  $1 - e^{-\Omega(\epsilon M)}$ ,  $\Pr[MH(A) = MH(B)] = \frac{|A \cap B|}{|A \cup B|} \pm \epsilon$

**PROOF.** We have  $\Pr[MH(A) = MH(B)] = \frac{|A' \cap B'|}{|A' \cup B'|}$ . Since, all coin tosses are independent,  $|A' \cup B'|$  is a sum of independent Bernoulli random variables, with expectation at least  $M$ . Sold by Chernoff bounds, the probability that it differs from the mean by more than an  $\epsilon$  fraction is at most  $e^{-\Omega(\epsilon M)}$ . Similarly the probability that  $|A' \cap B'|$  differs from its mean by more than  $\epsilon|\tilde{A} \cup \tilde{B}|$  is that most  $e^{-\Omega(\epsilon M)}$ , completing the proof  $\square$

## 4. TERM SIMILARITIES VIA LOCALITY-SENSITIVE HASHING

In this paper, we consider the relationship to be *asymmetric*. This is based on the number of co-occurrence of a term pair compared to the total number of occurrences of each term in a pair. As explained in Section 1, one of the terms could be a *hyponym* or *hypernym* w.r.t the other term.

We use sketching algorithms described in the previous section to estimate similarity wherein we use succinct sketches of objects in a collection to compute the similarity between objects. Objects  $x$  and  $y$  that are similar have  $sim(x, y) = 1$  while completely dissimilar objects would have  $sim(x, y) = 0$ , where  $sim(x, y) \in [0, 1]$  denotes the similarity between objects  $x$  and  $y$ .

Min-Wise Independent Permutations are useful sketching functions that allow us to compute symmetric similarities between sets in a collection of sets as defined in Equation 1. This measure of similarity is exactly the Jaccard Coefficient in information retrieval. In this analysis, we view document sets and terms interchangeably. We want to use a similarity function that supports the asymmetric relationship between terms in the collection. One such function is

$$sim_A(A, B) = \frac{|A \cap B|}{|A|}$$

This similarity measure can be interpreted as the natural conditional probability  $\Pr(B|A) = \frac{|A \cap B|}{|A|}$ ; that is, the probability of term  $B$  occurring in a random document given that it contains  $A$ . Given  $sim_S(A, B)$  (computed by Equation 1), one can compute  $sim_A(A, B)$  from basic set theory, by

$$sim_A(A, B) = \frac{sim_S(A, B)}{1 + sim_S(A, B)} \frac{|A| + |B|}{|A|}$$

Given a random collection of  $k$  locality-sensitive hash functions,  $MH_1, MH_2, \dots, MH_k$ , we can compute the sketch of a term by

$$S(t) = (MH_1(\mathcal{D}_t), MH_2(\mathcal{D}_t), \dots, MH_k(\mathcal{D}_t)),$$

where  $\mathcal{D}_t$  is the document set of term  $t$ . Now, given two terms  $t_1$  and  $t_2$ , we can compute the similarity between the

terms by using

$$\text{EstimateSim}_A(t_1, t_2) = \frac{|\{i \mid S(t_1)[i] = S(t_2)[i]\}|}{k}.$$

Since each coordinate of this sketch of two terms agrees with probability  $\text{sim}_A(t_1, t_2)$ , it follows from Chernoff bounds that the estimate is close to the desired similarity measure.

**THEOREM 4.** *With probability at least  $1 - e^{-\Omega(\epsilon k)}$ ,  $|\text{EstimateSim}_A(t_1, t_2) - \text{sim}_A(t_1, t_2)| \leq \epsilon$ .*

## 4.1 Higher-order similarity measures

Unsupervised term clustering algorithms have successfully used methods like LSI to compute clusters. There has been work to show that LSI exploits higher-order term co-occurrences [18]. We adopt a similar approach by considering a second order similarity computation. Using the term-term matrix containing the similarity between any two terms, we iterate over the same procedure to compute the second order similarity, this time using the set of similar terms,  $N_t$  for a given a term  $t$  instead of its document set as

$$\text{EstimatedSim}_{2A}(t_1, t_2) = \frac{|\{i \mid S_2(t_1)[i] = S_2(t_2)[i]\}|}{k},$$

where  $S_2(t) = \{MH_1(N_t), MH_2(N_t), \dots, MH_k(N_t)\}$ . Thus, we generate a term-term similarity matrix and the corresponding directed graph  $G = (V, E)$  where the weight on an edge  $(u, v)$  in  $G$  has weight  $w_{uv} = \text{EstimatedSim}_{2A}(u, v)$ . An edge is added to  $G$  only if  $w_{uv} \geq \theta$ , for some threshold  $\theta \in [0, 1]$ . In this study, we use  $\theta \in [0, 0.25]$ . We now present our algorithm for generating hierarchical topics based on the concept that two objects are similar if they share a lot of similar objects.

## 5. ALGORITHM FOR HIERARCHICAL TOPIC GENERATION

Given an  $m \times n$  term-document matrix  $A$ , we generate a hierarchical decomposition of the term space. Informally, our algorithm picks the top  $l$  terms in the collection of  $n$  documents. The reduced  $l \times n$  term-document matrix  $A'$  is used to generate the term-term similarities using LSH. The algorithm 3 takes a parameter,  $k$ , that specifies the number of hash functions to be used in the LSH scheme.

---

**Algorithm 3** GENERATE TOPICS( $\theta, l, A, N_h$ )

---

- 1:  $T \leftarrow t_0, t_1, \dots, t_l$  {top  $l$  ( $\ll m$ ) terms}
  - 2: Generate  $l \times l$  term-term matrix  $B$  s.t.  $b_{ij} = \text{EstimatedSim}_{2A}(t_i, t_j)$ . {not an all-pairs computation}
  - 3: Generate similarity graph  $G = (V, E)$  based on the underlying matrix  $B$ .
  - 4: Compute connected components  $T_1, T_2, \dots, T_p$  in  $G$  to generate the  $p$  topics.
- 

### 5.1 Time complexity analysis

Computing the top  $l$  terms takes time  $O(|M_{mn}|_{\bar{0}})$  where  $|M_{mn}|_{\bar{0}} (\ll mn)$  is the number of non-zero entries in the  $m \times n$  term-document matrix. Let  $|M_{ln}|_{\bar{0}}$  denote the number of non-zero entries in the truncated  $l \times n$  matrix. To compute term similarity, each  $MH$  computation takes time  $O(|M_{ln}|_{\bar{0}})$ . Assuming, on the average, most terms are dissimilar, populating the term similarity matrix takes  $O(l)$  per

iteration with a total running time of  $O(|M_{mn}|_{\bar{0}} + k|M_{ln}|_{\bar{0}})$ . From the guarantees of Theorem 3, for a confidence probability of  $1 - \delta$ , we need to set  $k = \frac{1}{\epsilon} \log(\frac{1}{\delta})$  giving a running time of  $O(|M_{mn}|_{\bar{0}} + \frac{1}{\epsilon} \log(\frac{1}{\delta})|M_{ln}|_{\bar{0}})$ . Compare this with the  $O(m^2n)$  running time for cosine similarity computation.

## 5.2 Remarks

### 5.2.1 Truncated term-document matrix

A simple analysis of the term-weight distribution yielded a heavy-tailed distribution with the number of terms with large weights being very small in the collection. Thus, reducing the dimensionality of the term space helps in 'filtering' spurious term relationships. Based on this observation, we compute the top  $l$  terms in a collection of documents as

$$w(t_i) = \frac{\sum_{j \in \mathcal{D}_i} \log(tf_{ij})}{|\mathcal{D}_i|} \log\left(\frac{N}{1 + |\mathcal{D}_i|}\right),$$

where  $N$  is the total number of documents in the collection and  $tf_j$  is the term frequency of term  $t_i$  in document  $j$ , and  $\mathcal{D}_i$  is the document set of term  $t_i$ . This weight is typically the average TFIDF score of the term. Thus, we compute a  $l \times k$  truncated version of the original  $n \times n$  term-term similarity matrix.

*Low rank approximation* - One practical application when dealing with large matrices is to find a low-rank approximation of the matrix and then work with the low-rank approximation to produce efficient algorithms. There is a lot of literature in the area of finding low-rank approximations [1]. Our approach is analogous to sampling entries in the term-document matrix with probabilities proportional to the weight of an entry in the matrix to generate a matrix of lower rank. An interesting open problem would be to characterize this truncated matrix as a valid low-rank approximation of the original term-term matrix. Another view that can be adopted would be to verify the properties of the reduced term similarity graph are good approximations to those in the original graph. Some examples of these properties could include a clustering of the original graph and commute times between any two nodes in the original graph.

### 5.2.2 Connected component analysis

Extracting topics in the term space is equivalent to computing the strongly connected components in the underlying reduced graph. In this study we use algorithms based on DFS to extract the SCCs in the directed graph. However, these algorithms can produce SCCs that represent topics that are not cohesive. By this we mean, the output could be weak in terms of the induced hierarchy of topics since the SCCs could be disconnected. To produce more granular hierarchies, it might be required to analyze individual components to extract sub-topics from each component and define the relationships between various sub-topics. One could employ algorithms to find directed sparse cuts to better analyze each SCC in the graph.

Flake, Tarjan, and Tsioutsoulouklis [10] have proposed a hierarchical graph partitioning algorithm using min-cut trees which is well suited for web and citation graphs. In this work, we adopt a simple intuitive approach to recursively partition a SCC in a directed graph. To generate the hierarchy, we generate the graph  $G$  in step 3 of Algorithm 3 with the threshold for edge weight,  $\theta$ , set to a very small value, say

0.01. We use a straight-forward DFS based scheme to partition the term similarity graph into forest of DAGs. Each DAG expresses the hierarchical relationship between topics. Any SCC in a DAG can be decomposed into a DAG of SCCs by setting the threshold parameter  $\theta$  to a larger value and recursively running our partition algorithm.

### 5.2.3 Symmetric similarity measures

One of the main advantages of using an asymmetric similarity measure is the resulting directed acyclic graph from any connected component analysis on the directed term similarity graph. Moreover, an asymmetric similarity measure better characterizes the relationship between any two terms. For example, if  $|A| = 100$  and  $|B| = 5$ , and  $|A \cap B| = 3$ , we have  $sim_S(A, B) = 0.30$ ,  $sim_A(A, B) = 0.3$ , and  $sim_A(B, A) = 0.6$ . If the threshold in a graph reduction step were set at 0.5, we would lose the relationship between  $A$  and  $B$  in the similarity graph when using  $sim_S(A, B)$ , whereas  $sim_A(B, A)$  would survive the graph reduction step.

## 6. EXPERIMENTAL EVALUATION

In this section, we detail the experiments we carried out to test the efficacy of our algorithms. We compare the term space decomposition to topic space produced by our algorithms with LSI. We report results that show that our algorithm, even with adopting simpler schemes for graph partitioning, produce accurate topic clusters. We test the algorithms with various values of parameters, *viz.* number of hash functions and the number of top weighted words in the collection. For lack of space, we do not report all results.

We tested our algorithms on a collection of 300 digital books with a resulting term-document matrix of size 20000 × 300. The resulting number of terms were arrived at by removing the stop words from the term set. The collection of books were picked to cover various topics like **health**, **computer networks**, **life sciences**, **music**, **defense**, etc. In addition to our specific collection, we ran the algorithms on standard collections like the open-source DMOZ project [13] to test the effectiveness of our similarity computations.

After generating the top  $k$  terms, we compute the term-document matrix where each entry in the matrix is the weight associated with a  $\langle term, document \rangle$  pair. For the case of Method 1, we use the TFIDF score of each term in a document as the corresponding matrix entry. For Method 2, we normalize this value using a large value for  $F = 100$ .

In this set of experiments, we set the top  $k$  terms used in the analysis a parameter of our algorithm. We varied  $k$  between 100 and 5000. We report results for  $k = 300$  and  $k = 1000$  in two sets of experiments we report. All experiments were run using Method 2 described in Section 3.

### 6.1 Hierarchical topic generation

The first set of experiments illustrate the hierarchical decomposition of the topic space given a term-document matrix. We ran the experiments on a collection of 300 documents representing **sports**, **life sciences**, **entertainment**, **economics**, and **communication networks**. Some documents in the collection included topics that belonged to more than one subject area. Table 1 shows the hierarchical decomposition of the topic space into the top 3 topics. For example, Topic 1 groups terms representing **religion**, **Eastern Europe**, **economics**, and **sports**. Some of the sub-topics produced under topic 1 are about **Eastern Europe**, **professional**

<p>Topic 1 (<b>Religion, E. Europe, Economics, Sports, Salaries</b>)</p> <p>1 - assemblies, solidarnosc, players, enduring, bonus, wfodarczyk, lockout, protestants, insurance, employee, czech, russian, calvinism, catholics, polish, polit, league, strike, nfl, communist, basketball, analyst, coalition, prescott, blessed, negotiations, euro, salary, malthusian, micro-economics, antitrust</p> <p>1.1 - bureaucracy, czech</p> <p>1.2 - russia, polit, nato, euro</p> <p>1.3 - calvinism, malthusian, protestants, catholics, profit</p> <p>1.4 - nba, nfl, bonus, players, basketball, strike</p> <p>1.4.1 - negotiations, salary</p> <p>1.4.2 - player</p> <p>Topic 2 (<b>Health, Supplements, Sports, Iraq, Immigration</b>)</p> <p>2 - myosin, insulin, supplements, poll, iraq, pedaling, latino, aerobic, minerals, anemia, zinc, hematuria, jobs, immigrant, endurance, musclem, mexican, armstrong, racing, cycling, anglo, voices, workforce, bartlett, diverge</p> <p>2.1 - myosin, muscle, mexican, muscular, pedaling, latino, minerals, supplements, armstrong, cycling, racing, poll</p> <p>2.1.1 - armstrong, pedaling, fiber, racing, exercising, myosin</p> <p>2.1.2 - training, bicyclist, exercising, surgeries, racing</p> <p>2.1.3 - vanadium, deficiency, hematuria, supplements</p> <p>2.1.3.1 - antioxidant, phosphate</p> <p>2.1.3.2 - deficiency, hematuria</p> <p>2.1.4 - iraq, mexican, latino, poll, voter, immigrant, rowman diverge, anglo</p> <p>2.2 - zinc, dietary, insulin, endurance, muscle, minerals</p> <p>2.2.1 - endurance, muscle, aerobic, cycling, physiological</p> <p>2.2.1.1 - anemia, magnesium</p> <p>Topic 3 (<b>Health, Networks</b>)</p> <p>3 - fibrosis, fault, bgp, nfs, bayes, multicast, disable, robustness, flood, psychosis, viruses, ring, tree, cache, bipolar, interfaces, shell, synchronization, tcp, hemisphere, topologies, tumor, attacker, sparse, handshake, metric, udp, carcinomas, overlay, syndrome, hbv, donor, transfused</p> <p>3.1 - ring, multicast, fifo, token, tree, successor, byzantine</p> <p>3.2 - fibrosis, carcinoma, neoplasia, invasive, surg, tumor, dysplasia, lesion, telomerase, suppressor, resection</p> <p>3.2.1 - surg, tumor</p> <p>3.3 - handshake, redirect, attacker, nfs, cache, icmp, queue, disable, flood, ack, timestamp</p> <p>3.3.1 - redirect, attacker, handshake, icmp, timeout, routed</p> <p>3.3.2 - flood, synchronization</p> <p>3.4 - constitution, parental, bipolar, recessive, allele, mitotic, cytogetic, psychosis, schizophrenia</p>
--

Table 1: Hierarchical topic decomposition

**sports**, **Christian theology**, **Polish politics**. One can intuit the relationship between **religion** and **Eastern Europe** when there is a talk of religious freedom and spread of religion in the erstwhile Soviet Union, **Eastern Europe** and **economics** of the Euro region, **sports** and **Eastern Europe** *vis-a-vis* the professional sports such as NHL in the United States, **sports** and **economics** under the subject of salary caps and such. Our algorithms effectively capture these relationships when run with small values of  $\theta$ . The values of  $\theta$  were increased to decompose a given topic into sub-topics. The values chosen were 0.1, 0.15, 0.18, 0.20, and 0.22. It is important to note the sub-topics are orthogonal to each other, very much like the term space decomposition produced by other principal component analyses. Our algorithms produced specific and orthogonal decompositions of other components as well.

**Effectiveness of our similarity computations** - We measured the effectiveness of locality-sensitive hashing to

extract similar terms. We use a “planted cluster” model, similar to the one described in [20], to test the effectiveness of our algorithms. In this model, we assign documents to pre-defined clusters and the accuracy of the generated topic space is measured by how much of the defined document clustering is recovered using the term sets in the topic space. We generate the document set,  $\{d_1, d_2, \dots, d_k\}$ , for the similarity computation from the DMOZ tree [13] using random sampling such that the document set is composed of distinct clusters. The correlation between topic  $\mathcal{T}_k$  and the document clusters is computed by considering all term pairs in a topic,

$$\phi_k = 1 - \frac{2}{|\mathcal{T}_k|(|\mathcal{T}_k| - 1)} \sum_{t_i, t_j \in \mathcal{T}_k} \sum_{C_i \in \mathcal{C}(t_i), C_j \in \mathcal{C}(t_j)} \frac{d(C_i, C_j)}{|C_i||C_j|},$$

where  $d(\cdot, \cdot)$  is a normalized distance function on the clusters. In the case of DMOZ hierarchy, it is the normalized path length between the roots of the clusters in the tree;  $\mathcal{C}(t)$  is a subset of the document clusters to which a term  $t$  belongs. One of the advantages of using such a correlation function is that it handles mis-classifications more gracefully. A term pair that is grouped in the same topic and yet has its terms belong to clusters far apart should have a very low correlation with the clusters. This value of correlation increases as the distance between the respective clusters becomes smaller and finally becomes 1.0 when both the terms in a term pair belong to a same cluster.

We set the number of clusters to 10. The size of the document set was set to 1000. Table 2 shows the topic clusters from the document set for  $\theta = 0.4$ . This value of the threshold produced 10 topic sets in the term space. A higher value of  $\theta = 0.5$  decomposed topic set 7 to produce content management and day commemorative, and century british among other topics. The last column in Table 2 shows the correlation between the topics and the planted clusters in the collection. Clearly, the topics correlate very well with the planted document clusters with  $\phi > 0.75$  for 80% of the topics.

## 6.2 Comparison with LSI

In the second set of experiments, we compare the topic space obtained by our algorithms with term-space decomposition obtained by LSI. We use MATLAB to carry out the term-document matrix decomposition for LSI. The truncated term-term matrix can be generated from the truncated term-document matrix using  $US^2U^T$ , where  $U$  and  $S$  are obtained by the singular value decomposition of the term-document matrix  $A = USV^T$ . To extract topics, we analyze the left-singular vectors, i.e., the columns of  $U$ . Some of the earlier work [5, 19] has proposed the relevance of the columns of  $U$  in understanding information retrieval concepts such as feature/topic selection and term similarities. A truncated SVD of the matrix  $A$  with  $k$  set to 10 produced 10 singular vectors corresponding to the top 10 eigenvalues of  $A$ . Along each dimension, we represent each topic by “grouping” the top weighted terms. We set the threshold for term selection to 0.1. Table 3 shows the term clustering produced by this method.

Clearly, one significant difference between the two clusterings is the presence of the same term (e.g., **banknotes**) with large weight in multiple clusters produced by the LSI based approach. Whereas, our approach was able to iden-

Topic	Terms
1	banknotes cornish anime
2	dsoverdragelser elizabeth builds erhvervsrelateret diskussionsforum byretter dcouvrir exposiciones
3	businesses drift bank clips byretter fanfics
4	dessinant driver enabling
5	dsoverdragelser enduring bustos
6	driver dessinant employment bars elments disorder
7	dessinant designs erhvervsret base byretter enteros entertainment erhvervsretlige abafil
8	designs erhvervsorienteret dessinant anime baserad byretter internet
9	erhvervsmssige burn drug banknote anim consulting bandes directory client efficient designed ability
10	animer software drug falukillar

Table 3: Clustering topics in DMOZ tree using LSI

Cluster	Keywords
1	banknote
2	credit card java
3	efficient business
4	brokers ability
5	available catalogs quality
6	utility
7	chinese bulk lists artifacts
8	emphasis dates canadian dollar collector
9	day comemorative
10	document bespoke services training company

Table 4: Keywords for the topics in the DMOZ tree

tify strongly correlated terms like **banknote** and **currency**, **credit** and **card**, and **commemorative** and **day**, LSI did not extract these relationships effectively. The clustering produced by LSI is more *soft* or *continuous* in nature as opposed to the discrete nature of the term clustering produced by our algorithms.

## 6.3 Computing keywords

It is often useful to have a succinct representation of a topic in the truncated term space. Keywords are one way to represent the information contained in a topic. Given a strongly connected component that represents a topic, the keywords can be associated with the “central” nodes in each SCC. These nodes are characterized by high out-degree. There are many ways one can compute the “centers” of such components. In our case where an edge between terms  $A$  and  $B$  implies  $A \prec B$ , it suffices to compute a vertex cover of the component. However, to ensure that the number of keywords is reasonably small compared to the number of terms in a topic, one simple approach would be to consider the top  $k$  terms in a component ranked by their weighted out degrees. Table 4 shows the keywords for each topic shown in Table 2. The analysis of a more formal approach to computing keywords in each topic is beyond the scope of this study. We point out that one could adopt approaches involving fixed-point solutions like the Pagerank and HITS algorithms [21, 16] to compute keywords in each topic.

## 7. CONCLUSIONS

We proposed algorithms based on LSH to efficiently com-

Topic	Terms	Cluster Correlation, $\phi$
1	banknote currency	1.00
2	credit card java cgi	0.60
3	efficient business enabling	0.92
4	brokers ability	1.00
5	available catalogs quality	0.58
6	utility applications dedicated	0.74
7	artifacts empire albums bronze world online collecting dealer coins chinese gold anglo states appraisal exnomia silver banknotes byzantine articles coin european bid english century catalog bulk mint dealing british contents books elongated antiquities auctions lists collectibles buying based dutch	0.86
8	dollar china emphasis circulated auction collection includes biblical canadian abafil dates collector assisting areas acquiring bars collectible	0.83
9	day commemorative	1.00
10	base corporate support application services data document commerce distribution management bespoke systems company distributed consulting provides content database development training exchange develops	0.79

Table 2: Clustering topics in the DMOZ tree

pute hierarchical relationships between terms in a given collection of documents. The algorithms use an asymmetric similarity measure between terms in a collection to compute a decomposition of the term space into a lower dimensional topic space from a term-document matrix. We provide both theoretical and experimental evidence to show the effectiveness and time efficiency of our algorithms.

There are interesting open questions that arise from this work. One open question is whether LSH can be shown to compute an effective low-rank approximation of the original term-document matrix. Another interesting problem would be to study effective graph partitioning algorithms to decompose a strongly connected component to generate sub-topics for the topic represented by the SCCT.

## 8. REFERENCES

- [1] Achlioptas and McSherry. Fast computation of low rank matrix approximations. In *ACM Symposium on Theory of Computing (STOC)*, pages 611–618, 2001.
- [2] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.
- [3] A. Budanitsky and G. Hirst. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In *Proc. of the North American Chapter of the Association for Computational Linguistics (NAACL-2001)*, Pittsburgh, PA, June 2001., 2001.
- [4] M. Charikar. Similarity estimation techniques from rounding algorithms. In *IEEE Symposium on Theory of Computing (STOC)*, pages 380–388, 2002.
- [5] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
- [6] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proc. 7th Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 269–274, 2001.
- [7] I. S. Dhillon, S. Mallela, and R. Kumar. Enhanced word clustering for hierarchical text classification. In *Proc. 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 191–200, 2002.
- [8] P. Drineas, A. M. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering large graphs via the singular value decomposition. *Machine Learning*, 56(1-3):9–33, 2004.
- [9] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.
- [10] G. W. Flake, R. E. Tarjan, and K. Tsioutsouliklis. Graph clustering and minimum-cut trees. *Internet Mathematics*, 1(4):385–408, 2004.
- [11] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *The VLDB Journal*, pages 518–529, 1999.
- [12] T. H. Haveliwala, A. Gionis, and P. Indyk. Scalable techniques for clustering the web. In *WebDB (Informal Proceedings)*, pages 129–134, 2000.
- [13] <http://www.dmoz.org>.
- [14] P. Indyk, R. Motwani, P. Raghavan, and S. Vempala. Locality-preserving hashing in multidimensional spaces. In *Proc. 29th ACM Symposium on Theory of Computing (STOC)*, pages 618–625, 1997.
- [15] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *Proc. 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 538–543, 2002.
- [16] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
- [17] J. M. Kleinberg and A. Tomkins. Applications of linear algebra in information retrieval and hypertext analysis. In *Proc. of the 18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 31 - June 2, 1999, Philadelphia, Pennsylvania*, pages 185–193. ACM Press, 1999.
- [18] A. Kontostathis and W. Pottenger. Detecting patterns in the LSI term-term matrix. In *Proc. Workshop on the Foundation of Data Mining and Discovery, IEEE International Conference on Data Mining (ICDM'02)*, 2002.
- [19] T. K. Landauer and S. T. Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104:211–240, 1997.
- [20] F. McSherry. Spectral partitioning of random graphs. In *IEEE Symposium on Foundations of Computer Science*, pages 529–537, 2001.
- [21] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.
- [22] L. Talavera. Feature selection as a preprocessing step for hierarchical clustering. In *Proc. 16th International Conf. on Machine Learning*, pages 389–397. Morgan Kaufmann, San Francisco, CA, 1999.