RC23023(W0312-030) December 5, 2003
Computer Science

IBM Research Report

Topical Document Clustering

Rie Kubota Ando
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Topical Document Clustering

Rie Kubota Ando
IBM T.J. Watson Research Center
19 Skyline Dr., Hawthorne, NY 10532
rie1@us.ibm.com

Abstract

We propose a novel document clustering method, which at a high level, seeks to uncover the topics underlying documents and to generate clusters accordingly. The process is driven by topical phrases. We rely on a subspace projection technique to find topical phrases and to measure how well those phrases describe the documents. Our evaluation on manually topic-labeled documents shows that the proposed method significantly outperforms all the tested methods. A useful side product of this method is a set of 'phrase trees', which can serve as cluster summaries when presented to users.

1 Introduction

Content-based document clustering has been studied for use in applications such as browsing search results and facilitating relevance feedback (Cutting et al., 1992; Hearst and Pedersen, 1996; Leouski and Croft, 1996; Schütze and Silverstein, 1997; Silverstein and Pedersen, 1997), distributed text retrieval (Xu and Croft, 1999), and topic detection (Franz et al., 2001).

The goal of our work is to develop a method to group documents in a way consistent with the underlying topics (recognized by humans) *without prior knowledge* of such topics. At a high level, this task can be described as finding the topics (or concepts) running through multiple documents and organizing the documents accordingly. In this paper we propose a new method which is a straightforward embodiment of this view, in which phrasal expressions serve as the representations of such topics. More specifically, the method selects and organizes phrasal expressions which convey the concepts prominently discussed in the corpus, and uses those phrases as the 'seeds' for document clustering.

In either unsupervised or supervised document classification, document representation is an important issue. While words are commonly used as document features (cf. Salton and McGill (1983)), more complex features, such as word clusters (Slonim and Tishby, 2000), have been also investigated. In particular, motivated by its clarity the use of phrases has been studied: syntactic phrases and phrase clusters for categorization (Lewis, 1992), combining linguistic features such as named entities for clustering (Hatzivassiloglou et al., 2000), statistical phrases for clustering (Zamir and Etzioni, 1998; Maarek and Wecker, 1994), and so on, reporting both positive and negative results. While we share the motivation, our approach differs in that phrases are *not features* of documents. Instead, we place phrases and documents in the same representation space, where we have reliable similarity measures between different types of text units. We exploit a subspace projection technique for such representation space; hence, we call our method SSP in this paper.

Latent Semantic Indexing (LSI) (Deerwester et al., 1990) is a well-known method to represent text units via subspace projection. In essence, LSI expands the classical *Vector Space Model (VSM)* vectors with related words, where the word relations are automatically (and implicitly) derived from the word cooccurrences observed in the documents. Ando and Lee (2001) formally showed that LSI's

performance depends on the document distributions over (hidden) topics[1], and consequently, proposed a generalization of LSI, the *Iterative Residual Rescaling (IRR)* method. It was experimentally shown that IRR document representation produce document clustering performance superior to that of LSI and VSM, especially when documents are less uniformly distributed. Use of IRR for phrase representation was suggested but has never been evaluated.

In this work, we use the IRR algorithm for representing documents and phrases in a vector space. Through experiments with variations of SSP, we show that phrases are useful for document clustering when combined with similarity measures based on IRR vector representation.

We start with the review of the IRR method below, and present our SSP method in Section 2. Section 3 describes our experimental framework, and introduces the methods which we tested for comparison. We report and analyze the evaluation results in Section 4, and conclude in Section 5.

1.1 Iterative Residual Rescaling (IRR)

IRR is a method to generate vector representations for text units. As in LSI, the essence of IRR is to expand classical VSM vectors with the related words, where the word relations are automatically derived from the word cooccurrences observed in the documents. It captures the word relations into an *IRR subspace*, and produces *IRR vectors* via subspace projection. The input to the IRR subspace computation is a parameter $q(\geq 0)$ called *scaling factor* and the VSM document vectors $\mathbf{d}$'s whose elements are associated with word frequencies. The scaling factor q relates to the degree of counteracting non-uniformity of topic-document distribution. The IRR basis vectors $\mathbf{b}_j$'s (which define the subspace) are computed to satisfy the following recursive conditions:

$$\mathbf{r}_i^{(0)} = \mathbf{d}_i, \qquad \mathbf{b}_j = \arg\max_{|\mathbf{x}|=1} \sum_{i=1}^{n} |\mathbf{r}_i^{(j)}|^q (\mathbf{r}_i^{(j)^T} \mathbf{x})^2, \qquad \mathbf{r}_i^{(j+1)} = \mathbf{r}_i^{(j)} - \mathbf{b}_j^T \mathbf{r}_i^{(j)} \mathbf{b}_j$$

The IRR vector $\hat{\mathbf{d}}$ for a VSM vector $\mathbf{d}$ is the projection of $\mathbf{d}$ onto the subspace spanned by $\mathbf{b}_j$'s: $\hat{\mathbf{d}} = \sum_j (\mathbf{b}_j^T \mathbf{d}) \mathbf{b}_j$.

Ando and Lee (2001) showed that LSI is likely to produce poorer document representations when the topic-document distribution is less uniform. This theoretical expectation was experimentally validated, and it was shown that IRR with a large scaling factor q can counteract large non-uniformity of the distribution – thus, outperforming LSI.

The effectiveness of using IRR in the SSP method is experimentally validated in comparison with the alternatives in Section 4.

Notational conventions: Let us fix the set of documents to be processed and denote it by D. we let n and m be the number of documents and words in D, respectively. We use $\mathcal{C}$ to denote an arbitrary *partition* of D. A partition is a collection of sets of documents, and precisely covers D. We use c denote an arbitrary element of a partition – that is, c is a set of documents – and call it *cluster* c.

[1] We say the *topic-document distribution is uniform* if, for example, half of the documents discuss one topic and the other half discuss another topic. We say it is *less uniform* if, for example, 75 documents are on one topic and 25 documents are on another.

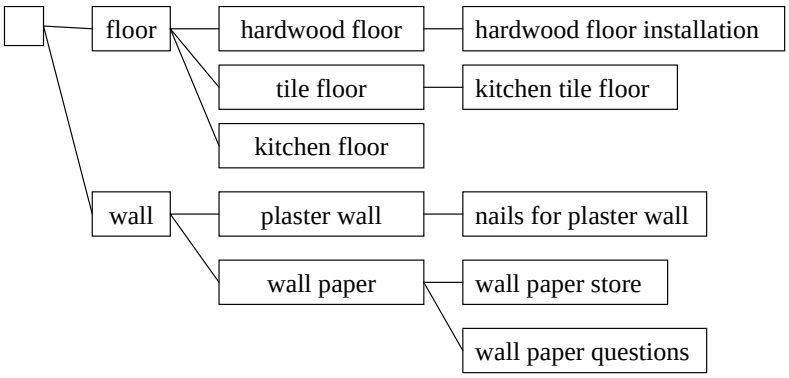


Figure 1: An example phrase descriptor tree.

2 The SSP document clustering method

The basic idea is to generate clusters and succinct cluster descriptors (or cluster labels) *simultaneously*. Descriptor candidates (which are syntactic phrases extracted from the documents) drive the clustering process; thus, we call our method phrase-driven document clustering. Note that this is in contrast to typical approaches where the most discriminating features are chosen as labels in the end of clustering process. A rough sketch of the phrase-driven process is as follows. Suppose that some partition $\mathcal{C}$ is given as input. For each cluster c in $\mathcal{C}$, we choose a few phrases that describe the major topics discussed in c. Some documents may not be described by the chosen descriptor phrases since we want only a limited number of short descriptors. Now, suppose that we fix the descriptors, and try to change $\mathcal{C}$ to make more documents fit their cluster descriptors. If we see descriptors like 'theater play' and 'export license' for one cluster c, we split c into two clusters. But if they are 'nuclear weapon' and 'chemical weapon', we may want to leave the cluster as it is. If we find that some document in c is better described by a descriptor for another cluster c', we move that document from c to c'. We call these operations phrase-driven *split* and *update*.

The method has two more operations, *merge* and *estimation*. Their roles are to control the number of clusters and choose the best partition. We start with a randomly-generated partition of the desired number[2] of clusters. We refine the clusters by repeatedly splitting, updating, and merging, while estimating the quality of the partition. After some fixed number of iterations, we choose the partition estimated to be the best as the final result. Intuitively, split/update focuses on 'purifying' the clusters, and the merge process puts emphasis on improving topic integrity. The expectation is that while 'shuffling' the documents by repeatedly splitting and merging, a reasonable partition should emerge and be noted by the estimator. We describe the operations precisely below.

Let us assume that words and phrases have been extracted by linguistic processing prior to clustering, and that we have the word-document, word-phrase, and phrase-document matrices of the frequencies of smaller text units in larger text units. In this method, a *phrase* is a syntactic phrase such as a noun phrase, which may be either single-word or multi-word expression.

Subspace-based measures We define three types of measures based on IRR. To facilitate presentation, we delay their precise definitions to Section 2.5, and here describe their essence.

[2]While the automatic determination of the optimum number of clusters is an important issue, we consider that in practice clustering methods should have the capability to generate the specified number of clusters. This number may come from the desired granularity or from the constraint of the presentation scheme for browsing the results.

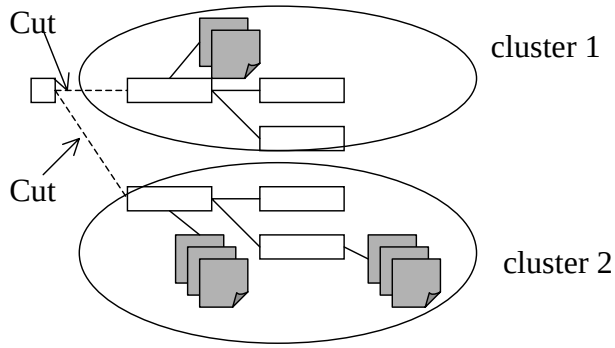


Figure 2: Cutting a phrase descriptor tree to split clusters

- *Topicality* of phrase p with respect to c measures the degree to which phrase p conveys the topics discussed in a cluster c, and is denoted by $\tau_c(p)$.
- *Relation measure:* We let $\rho_c(x, y)$ denote the relation between text units x and y measured by taking account of word cooccurrences in cluster c.
- *Subspace-based cluster similarity* is the similarity between cluster c and c' measured by comparing the word cooccurrences in c and c', and is denoted by $\sigma(c, c')$

2.1 Phrase-driven update

The phrase-driven update is similar to one iteration of the standard K-means algorithm except that we have 'descriptor phrases' instead of the centroid of a cluster. Let $\mathcal{C}$ be the input partition. We assign *descriptor phrases* to every $c \in \mathcal{C}$. The descriptor phrases of c are the phrases that occur in the documents in c and that have the highest topicality τ_c. We set the maximum number of descriptor phrases per cluster to some constant (e.g., 100). Next, for every $c \in \mathcal{C}$, we measure the phrase-document relation $\rho_c(p, d)$ for every descriptor phrase p for c and every document d in D. We move document d to the cluster with the descriptor phrase most tightly related to d. Note that for c's descriptor phrases, the phrase-document relation is measured considering word cooccurrences in c.

2.2 Phrase-driven split

Given a partition $\mathcal{C}$, we choose the descriptor phrases for every $c \in \mathcal{C}$ and associate each document with its most related descriptor phrase, exactly in the same manner as in the phrase-driven update. For every $c \in \mathcal{C}$, we construct a phrase descriptor tree by choosing a parent descriptor phrase for each descriptor phrase p as follows:

- The constituent words of a parent phrase must be a subset of those of p. For instance, "floor" or "hardwood" can be a parent of "hardwood floor" but not vice versa. If p does not have any legal parent phrase, we connect it the empty root node.
- A phrase having more words (therefore, overlapping more with p) is preferred, i.e., "hardwood floor" is preferred to 'floor' as a parent of "hardwood floor installation".
- Ties are broken by preferring the phrase whose topicality τ_c is closer to that of p. Further ties are broken arbitrarily.

Input: the number of clusters K, the set of clusters $\mathcal{C}_{old}$
Output: New set of clusters $\mathcal{C}_{new}$

$\mathcal{C} := \mathcal{C}_{old}$
For every $x, y \in \mathcal{C}$, $\mathrm{sim}[x, y] := \sigma(x, y)$.
$\mathcal{C}_{new} := \{\}$
loop
  for $c \in \mathcal{C}$ s.t. $|c| = \min_{x \in \mathcal{C}} |x|$ // For all the smallest clusters,
    $c' := \arg\max_{x \in \mathcal{C}} \mathrm{sim}[c, x]$ // choose the closest.
    $\hat{c} := c' \cup c$ // c' absorbs c
    For every $x \in \mathcal{C}, \mathrm{sim}[\hat{c}, x] := \mathrm{sim}[c', x]$
      // do not recompute similarities
    $\mathcal{C} := \mathcal{C} - c - c' + \hat{c}$ // Replace c and c' with $\hat{c}$
  if ($|\mathcal{C}| < K$) // No more merge is necessary.
    leave loop
  $\mathcal{C}' := \mathrm{C\_Link}(\mathcal{C}, \mathrm{sim}, K)$ // Apply complete-link to generate K clusters.
  if $\mathcal{C}_{new}$ is empty or $\mathcal{C}'$ is estimated to be better than $\mathcal{C}_{new}$
    $\mathcal{C}_{new} := \mathcal{C}'$ // Keep the best.

Figure 3: Merging clusters

In the resultant tree the phrases become more specific towards the leaves due to the effect of word set inclusion, and those in the same sub-tree are somewhat related to each other as they share some words and have similar topicality. An example of a phrase tree is shown in Figure 1. Finnaly, we 'cut' the trees at the root to generate new document clusters, as in Figure 2.

2.3 Merge

The input to the merge process is K, the desired number of clusters, and a partition of size h such that $h > K$. To generate K clusters from h clusters, we employ simple heuristics and a standard clustering algorithm. Since a very small cluster is unlikely to be an individual cluster in the optimum partition, we let the smallest clusters absorbed by other clusters first, and then we apply the standard complete-link algorithm to the resultant clusters, using the subspace-based cluster similarity $\sigma(c, c')$. More precisely, we initialize the work cluster set $\mathcal{C}$ with the given h clusters. We let each of the smallest clusters in $\mathcal{C}$ absorbed by its closest cluster, thereby changing $\mathcal{C}$. We apply the complete-link algorithm to $\mathcal{C}$ to obtain K clusters. This process is repeated until the number of clusters in $\mathcal{C}$ becomes less than K. Each time we apply the complete-link, we estimate the quality of the obtained partition using the estimation function defined in the next section. The one estimated to be the best is the output of the merge process. For efficiency, we compute the cluster similarities once at the beginning of the merge, and do *not* recompute them when cluster c' absorbs c. High-level pseudo code for merge is found in Figure 3.

2.4 Partition quality estimation:

We estimate the quality of partition $\mathcal{C}$, considering the 'tightness' of each cluster and the 'separation' between the clusters as follows:

$$\text{quality}(\mathcal{C}) = \sum_{c \in \mathcal{C}} \frac{|c|}{n} \cdot \frac{g(c,c)}{g(c,c) + \left(\max_{x \in \mathcal{C}, x \neq c} g(c,x) + g(D,D)\right)/2}, \text{ where } g(c,c') = \sum_{\mathbf{d} \in c, \mathbf{d}' \in c'} \frac{\mathbf{d}^T \mathbf{d}'}{|c| \cdot |c'|}$$

As is commonly done in the standard group average algorithm, the closeness of two clusters $g(c,c')$ is defined to be the average of the inner products of document pairs spanning the two clusters. The expectation is that a larger $g(c,c)$ indicates c is purer, and that a smaller $\max_{x \in \mathcal{C}, x \neq c} g(c,x)$ indicates c is well-separated from the other clusters. The average inner products $g(D,D)$ in the denominator is necessary to assign low scores to partitions in which a single cluster has almost all documents. Such partitions are likely to be poor.

2.5 Using IRR

SSP computes the IRR subspaces from subsets of documents instead of the entire set. This differs from the previous LSI work such as (Deerwester et al., 1990; Wolfe et al., 1998), which computed the subspace from all the documents at once. More specifically, we *recompute the subspace for each cluster*, whenever we change the partition by split/update/merge. By doing so, we recapture the word relations representing the topic(s) carried by that cluster. (In addition, this approach saves computation time, due to the quadratic property of subspace computation. See Section 3.5 for runtime considerations.)

Since we compute multiple subspaces, for clarity we let $\mathcal{X}_c$ denote the IRR subspace computed from the documents in cluster c, and let $\mathbf{B}_c$ denote the set of basis vectors spanning $\mathcal{X}_c$. We define three types of measures used in SSP as follows.

- *Relation measure:* We measure the relation between text units by the inner product of the corresponding IRR vectors. More precisely, we define $\rho_c(x,y)$ to be the inner product of the IRR vectors that are generated via projection onto $\mathcal{X}_c$.
- *Topicality:* The *topicality* of a phrase p with respect to c is an approximation of the sum of $\rho_c(p,d)^2$ over all the documents in c: $\tau_c(p) = \sum_{\mathbf{b} \in \mathbf{B}_c} (\mathbf{p}^T \mathbf{b})^2 \sum_{\mathbf{d} \in \mathbf{C}} (\mathbf{d}^T \mathbf{b})^2$, where $\mathbf{C}$ is a set of VSM document vectors for c. The approximation is necessary for efficient computation.
- *Cluster similarity:* Since we expect that $\mathcal{X}_c$ captures the word relations specific to the topics discussed in c, we also expect that the degree of 'overlapping' of $\mathcal{X}_c$ and $\mathcal{X}_{c'}$ should indicate the similarity between cluster c and cluster c'. We define the *subspace-based cluster similarity* to be the average of the sum of squares of inner products between the basis vectors of the two subspaces: $\sigma(c,c') = \sum_{\mathbf{b} \in \mathbf{B}_c} \sum_{\mathbf{b}' \in \mathbf{B}_{c'}} (\mathbf{b}^T \mathbf{b}')^2 / (|\mathbf{B}_c| \cdot |\mathbf{B}_{c'}|)$.

3 Experimental framework

To measure the degree to which the partition proposed by the method is consistent with the topics recognized by humans, we applied the method to the documents judged as relevant to one of the TREC topics. The task resembles the categorization task except that the method is *not* given any prior knowledge of the categories (which are TREC topics in this setting) other than the number of the categories (given as the desired number of clusters).

	topic 1	topic 2	topic 3	topic 4	topic 5
cluster 1	2	10	**15**		
cluster 2	3	10	5		
cluster 3				**20**	15
cluster 4	**15**				
cluster 5					5

Figure 4: Sample contingency table Z, with u(Z) = (15 + 15 + 20)/100 = 50%. cluster purity $\tilde{\mathrm{p}}(Z) = (15+10+20+15+5)/100 = 65\%$, topic integrity $\tilde{\mathrm{r}}(Z) = (15+10+15+20+15)/100 = 75\%$.

3.1 Evaluation metric

As in (Ando and Lee, 2001; Slonim and Tishby, 2000), we use a contingency table to measure the consistency between the proposed partition and the human-annotated topic labels. Let Z be a cluster-topic contingency table such that $Z[i, j]$ is the number of documents in cluster i that are relevant to topic j. We define the *unique-max rate* $\mathrm{u(Z)} = \sum_{\mathrm{i,j}} \mathcal{N}_{\mathrm{ij}}/\mathrm{n}$, where n is the number of documents and

$$\mathcal{N}_{ij} = \begin{cases} Z[i,j] & \text{if } Z[i,j] \text{ is the } \textit{unique} \text{ maximum in both its row and column} \\ 0 & \text{otherwise} \end{cases}$$

Note that this (rather strict) measure only considers the most tightly coupled topic-cluster assignments, and decreases when either 'cluster purity' or 'topic integrity' falls (see Figure 4).

For analysis, it is convenient to formalize the notion of cluster purity and topic integrity. We define *cluster purity* as: $\tilde{\mathrm{p}}(Z) = \sum_i \max_j Z[i,j]/n$, and *topic integrity* as: $\tilde{\mathrm{r}}(Z) = \sum_j \max_i Z[i,j]/n$. The cluster purity (and topic integrity) corresponds to microaveraged precision (and recall) in categorization evaluation, respectively (cf. Lewis, 1992). However, they are less strict in the sense that they consider one-to-many (and many-to-one) topic-cluster assignments, respectively, whereas unique-max rate considers only one-to-one assignments. As easily shown, $\mathrm{u(Z)} \leq \min(\tilde{\mathrm{p}}(\mathrm{Z}), \tilde{\mathrm{r}}(\mathrm{Z}))$.

3.2 Test data

We created two types of test set collections, which we call controlled-distribution collection and search-simulation collection, respectively.

To study the performance dependencies on the document distributions over the topics, we fixed the ten TREC topics (thus reducing the effects from the factors other than the distributions), and generated 30 sets each of which consists of 500 documents. These sets had six distribution types, five document sets each, in which nine (out of ten) topics have the same number of documents. We let the populations of those nine topics range from 25 to 50 by an increment of 5, thus generating six types.

To evaluate the performance on the document sets which we may encounter in practice, we generated 50 test sets by simulating search operations. First, we arbitrarily chose k topics, and from all the documents relevant to the chosen k topics, we selected 500 documents which are the most relevant to a certain word. The relevance was measured by the cosine between the corresponding VSM vectors, which were weighted by the standard tf-idf and were length-normalized. This simulates a simple vector-based search operation. We repeated it using 10 distinct words (simulating ten queries) for the number of topics $k = 10, 15, 20, 25, 30$, we obtained 50 test sets. The resultant sets had a wide variety of topic-document distributions.

3.3 Comparison with other algorithms

We studied the performance of the SSP method in comparison with the following methods: the *sequential Information Bottleneck (sIB)* algorithm (Slonim et al., 2002), standard clustering algorithms (complete-link, K-means, and group average) (see e.g., Manning and Schütze (1999)) using LSI, IRR (with several values of the scaling factor), and VSM to generate vector representations.

The sIB algorithm: Given the number of clusters K, sIB starts with randomly generated K clusters, and greedily move documents from one cluster to another to maximize the mutual information between the clusters c and words w: $I(\mathcal{C};W) = \sum_{c\in\mathcal{C},w\in W} p(c)p(w|c)\log\frac{p(w|c)}{p(w)}$. This (local) maximization, guaranteed to converge, is repeated t times from t distinct random seeds, and the cluster partition which gives the largest mutual information is chosen as the final result. Slonim et al. (2002) showed consistent superiority of sIB over several algorithms – including the agglomerative version of sIB (*AIB*), Iterative Double Clustering (El-Yaniv and Souroujon, 2001) which is an extension of AIB, and K-means and its variations.

LSI- and IRR-based clustering: The IRR-based clustering computes the subspace from all the documents to be clustered and applies the complete-link (or other standard algorithms) using the cosine between the IRR document vectors as the similarity measurement. The LSI-based clustering is a special case of the IRR-based with the scaling factor at 0. Ando and Lee (2001) studied their performances to evaluate the IRR-based similarity measurement, and showed superiority of IRR over LSI and VSM.

3.4 Implementation

As preprocessing, we extracted the stemmed nouns, proper nouns, and adjectives from the documents, after disambiguating parts-of-speech, and removed standard stop-words. We extracted noun phrases with (and without) an immediately-following prepositional phrase, using a syntactic parser (Boguraev and Neff, 2000). The choice of the types of parts-of-speech and phrases was based on the assumption that nominal expressions are sufficient to recognize the topics. For instance, for a miniature example document "He was the president of ABC Computer.", the set of extracted words would be { "president", "abc", "computer"}, and the set of extracted phrases would be {"the president", "the president of ABC Computer", "ABC Computer"}, respectively. Phrases are normalized into the sets of words, i.e., {{"president"}, {"president, "abc", "computer"}, {"abc", "computer"}}. The document vectors were length-normalized word frequencies for the subspace computation and the estimation in SSP, and the standard tf-idf term weighting was applied otherwise. This was done for all the methods.

To use IRR in SSP, we need to specify two parameters: the dimensionality of the subspace and the scaling factor. Following the original work, we choose the dimensionality k based on the *residual ratio* (the proportion of the document vectors left out from the subspace) with the constraint that $k \leq K$. We fixed the scaling factor and the residual ratio to 4 and 0.75, respectively, based on the observation on the document sets disjoint from the test data, and used them for all. We did not attempt the automatic scaling factor determination method.

The number of iterations was set to 10. As we will see later, fewer iterations can produce almost equivalent performances.

3.5 Runtime considerations

SSP's subspace computation runs in $O(n \cdot \min(m, n))$, regarding the desired number of clusters K and the average document length as constants. When there are many phrases, the selection of descriptor phrases and the document assignment become the most expensive operations. Our implementation considers only a fixed number (= 10) of phrases per document, while leaving out those having lowest topicality with respect to that document. As described above, we choose up to a constant number (= 100) of descriptor phrases per cluster. Therefore, both operations run in $O(n \cdot k)$ where k is the number of clusters, and $K \leq k << n$.

3.5.1 Implementation of other algorithms

The implementation of sIB followed the original work (Slonim et al., 2002) including the selection of 2000 words which had the most contribution to the mutual information. We noted that this word selection process was necessary in order to achieve reasonable computation time. The only deviation was the starting word set for the word selection. We tested the set used for SSP and others (part-of-speech filtered stemmed words) and the lower-case unstemmed tokens as in the original work. We only report the results using the former since it generally achieved better performances. We set the number of trials to 10 and the maximum number of iterations to 30. We observed that it always converged before iterating 30 times.

For the IRR- and LSI-based clustering, we set the dimensionalities of the subspaces to the number of topics as in one of the settings in (Ando and Lee, 2001). For the clearer comparison, we did not attempt the automatic determination of the scaling factor; instead, we tested three values of the scaling factor: $q = 4, 8$, and 12.

4 Experimental Results

The tested methods are SSP, sIB, and three standard algorithms (complete-link, K-means, and group average) using the LSI, IRR, and VSM vectors. For simplicity, we omit the results of K-means and group average, which generally underperformed the others. We denote the LSI-, IRR-, and VSM-based complete link by LSI-C, IRR-C, and VSM-C, respectively. The number of clusters was set to the number of the topics in every run.

In all the figures, performance metric is the unique-max rate (%) unless specified otherwise.

4.1 Results

We first examine our results on the controlled-distribution collection. Figure 5 plots the results average over five data sets of the same topic-document distribution types. We see that SSP is insensitive to the non-uniformity of topic-document distributions, and outperforms all baselines except sIB on the completely uniform distribution. In contrast, sIB and LSI-C are clearly vulnerable to non-uniformity while they rival SSP on uniform distributions. The IRR-C's trends are in line with the previous study, i.e., with smaller scaling factors it becomes vulnerable to non-uniformity (e.g., see $q = 4$ of Figure 6 (b)). The 'error bar' for SSP is one standard deviation over five runs with different initial random partitions. The small deviations indicate that SSP is insensitive to the changes in the initial partitions.

Figure 6 shows the performance on the search-simulation collection in relation to the number of topics. Each plot is the average over 10 data sets. For all the methods, not surprisingly, the performance

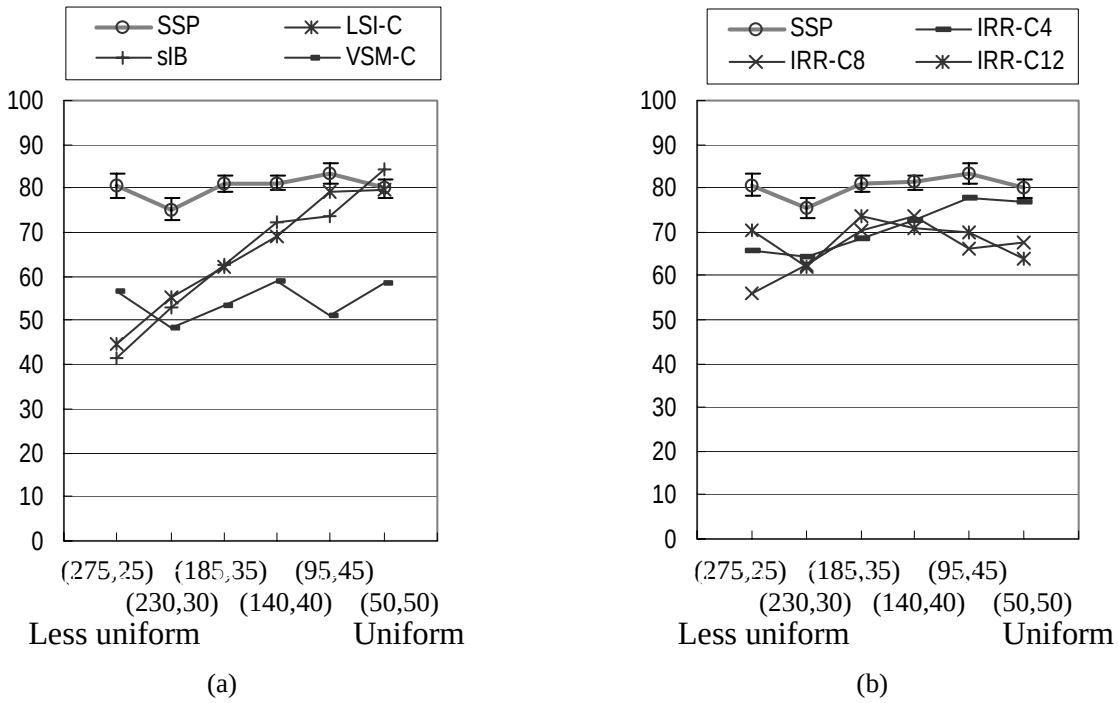


Figure 5: Results on the controlled-distribution data (10 topics) in comparison with: (a) LSI-C, sIB, and VSM-C, and (b) IRR-C (q=4, 8, 12). "(i,j)" indicates that one topic has i relevant documents and each of the other nine topics has j. Each plot is the average over five data sets.

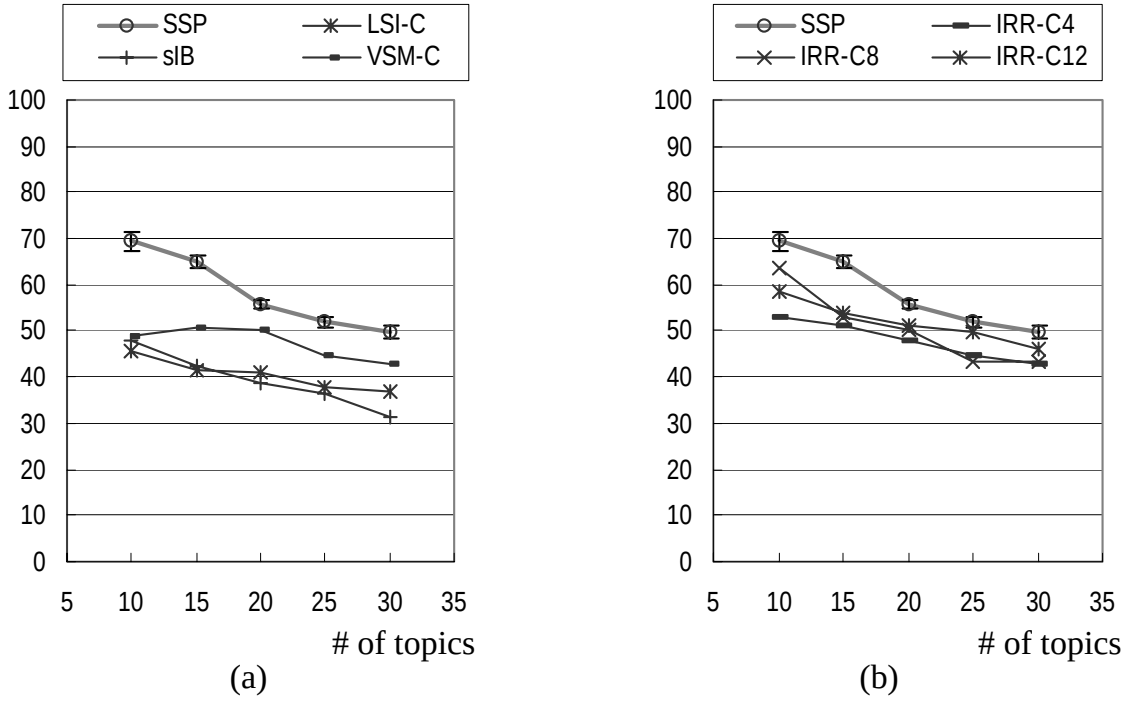


Figure 6: Search-simulation collection results in comparison with: (a)LSI-C, sIB, and VSM-C, (b) IRR-C (q=4, 8, 12). The x-axis is the number of the topics. Each plot is the average of 10 data sets.

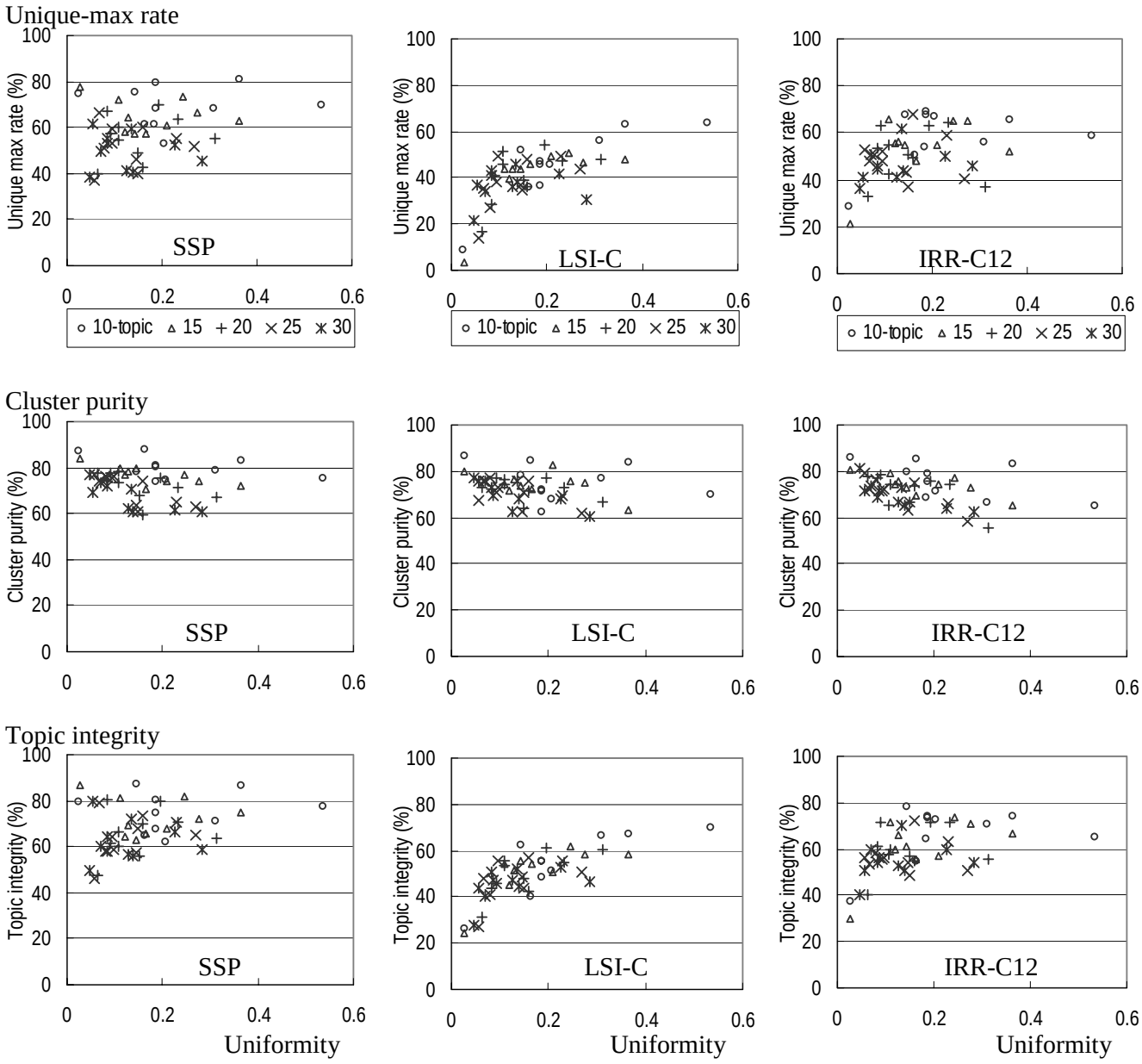


Figure 7: unique-max rate, cluster purity, and topic integrity on the search-simulation collection. SSP, LSI, and IRR-C($q = 12$) – the best-performing baseline – are shown. The x-axis is the *uniformity* of the data.

	Topics 1 – 10									
Cluster 1	**66**	1								
Cluster 2	7	**22**		20	1		19			
Cluster 3	60									
Cluster 4	57						1			
Cluster 5			2		**20**				1	
Cluster 6	18			1	3	**25**	2			
Cluster 7	33						1			
Cluster 8		2	23	4	1			**25**	24	
Cluster 9	34									
Cluster 10							2			**25**

Figure 8: Contingency table Z for LSI-C's result on a particular test set. unique-max rate u(Z) = (66 + 22 + 20 + 25 + 25 + 25) / 500 = 36.6 %, cluster purity $\tilde{p}(Z) = 73.4$ %, topic integrity $\tilde{r}(Z) = 53.8$ %.

	Topic 1 – 10									
Cluster 1	**254**					1	2			
Cluster 2		**23**			1					
Cluster 3			**18**		1			1		
Cluster 4	1			**24**			1			
Cluster 5	4		7	1	**23**	1	1	2	5	
Cluster 6	5					**23**		1	1	
Cluster 7	5	2					**21**			
Cluster 8	5							**21**		
Cluster 9	1								**19**	
Cluster 10										**25**

Figure 9: Contingency table Z for SSP's result on the same test set as in Figure 8. u(Z) = $\tilde{p}$(Z) = $\tilde{r}$(Z) = 90.2%.

drops as the number of topics increases, reflecting the difficulty of the task. Again, SSP generally outperforms the others.

4.2 Cluster purity and topic integrity

Let $t_1 \geq t_2 \geq \ldots \geq t_K$ be the numbers of documents relevant to the K topics in a data set. We formalize the notion of *uniformity* as: $\frac{t_2+\ldots+t_k}{K-1} \cdot \frac{1}{t_1}$. The uniformity ranges in $(0, 1]$, reaching 1 when (but not only when) all the topics have the same population size. Figure 7 plots the unique-max rate(first row), the cluster purity (second row), and the topic integrity (third row) of all the 50 sets in the search-simulation collection. The x-axis is the uniformity of the data sets. We observe that LSI-C's topic integrity degrades when the uniformity is smaller, which degrades the total performance unique-max rate (Recall that unique-max rate is no greater than the minimum of the cluster purity and the topic integrity.) In contrast, its cluster purity shows no dependency on uniformity, and rivals SSP. We have observed that sIB's cluster purity and topic integrity show a very similar trend as LSI-C. In contrast, SSP does not have clear dependency on uniformity.

Figure 8 shows the partition that LSI-C produced from one of the less uniform data sets in the

	Descriptor	Tree?	relation measure	Cont	Search
SSP	phrases	Yes	IRR	81.8	58.2
P-SSP-flat	phrases	No	IRR	-0.9	+0.2
W-SSP	words	No	IRR	-1.1	-2.5
P-VSM	phrases	Yes	VSM	-10.1	-4.9
P-VSM-flat	phrases	No	VSM	-15.4	-11.2
W-VSM	words	No	VSM	-16.8	-10.8
IRR-C4, IRR-C12	–	–	–	-10.9	-6.3

Figure 10: Performances of the variations relative to the SSP. The last row shows the best baselines, for comparison. 'Cont': average over 30 sets in controlled-distribution collection, 'Search': average over 50 sets in search-simulation collection.

controlled-distribution collection. In this partition, the most dominant topic is essentially split into five clusters (Cluster 1, 3, 4, 7, and 9), and Cluster 2 and 8 are a mixture of three topics. Considering only the most tightly coupled topic-cluster assignment (highlighted in the figures), LSI-C essentially detects six topics out of ten from this data set, whereas SSP detects all (Figure 9). This LSI-C's trend would be harmful for use in relevance feedback, where relevant documents may not be the majority of the search result.

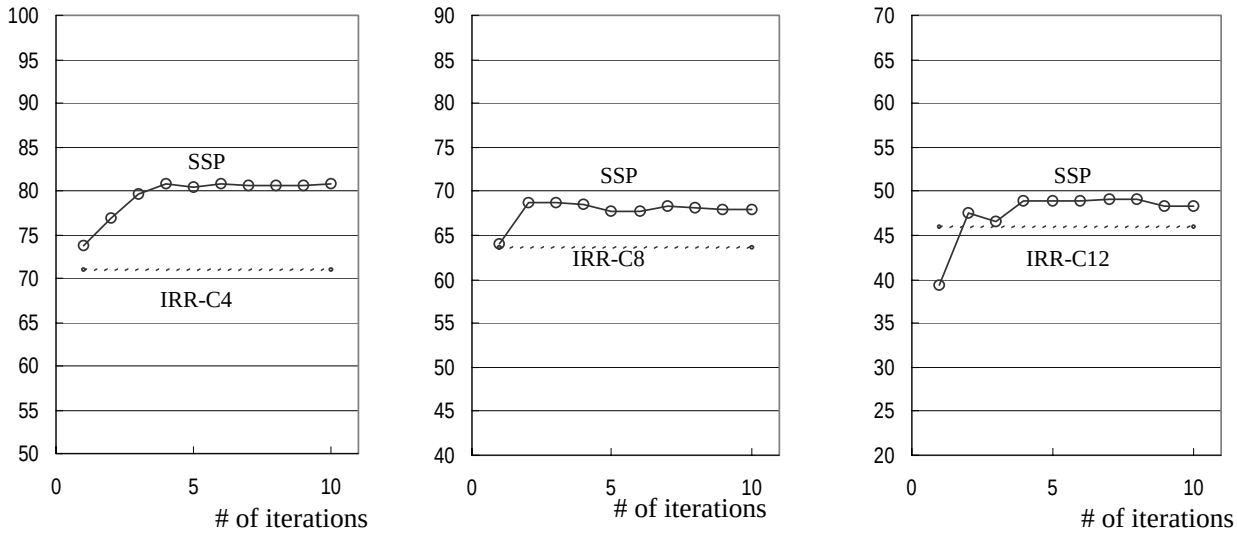
4.3 The effects of the phrase tree and IRR relation measure

To study the effects of using the phrase tree and IRR in the phrase-driven split, we examined the following five variations of SSP: P-SSP-flat links all the phrases to the root; P-VSM measures the phrase-document relations by the cosine of the corresponding (tf-idf weighted) VSM vectors instead of using IRR; W-SSP uses only the single-word expressions as descriptors; W-VSM uses only the single-word expressions, otherwise same as P-VSM. Figure 10 summarizes the variations and their performances on average over all the 30 test sets of the controlled-distribution collection (left column) and over all the 50 test sets of our search-simulation collection (right column). The initial randomized partitions were the same for all the methods.

We see that the use of IRR has a clear contribution. P-SSP-flat and W-SSP (both using IRR) rival SSP and significantly outperform all the three VSM-based variations. When the relation measure is IRR, the contribution of phrase trees (observed in SSP vs. W-SSP comparison) is not large but recognizable. The effect of the tree alone (SSP vs. P-SSP-flat) is less clear. In contrast, when the relation measure is VSM, the effectiveness of phrase trees (P-VSM vs. W-VSM) and the tree alone (P-VSM vs. P-VSM-flat) stands out. As a whole, we see that the best-performing baseline (in the last row of Figure 10) is outperformed when and only when the variation used *either IRR relation measure or phrase trees*. That is, phrases significantly contribute to document clustering *only when relationships among phrases are adequately represented* and used by the process – either explicitly via phrase tree generation or implicitly via phrase-document relation measure via IRR vector representations.

4.4 Iterations

Recall that SSP iteratively splits and merges the clusters. Figure 11 plot the performances of SSP against the number of iterations. The values are the average over: (a) 30 test sets of the controlled-distribution collection, (b) ten 10-topic sets and (c) ten 30-topic sets of the search-simulation collection.



(a) controlled-distribution sets (b) search-simulation 10-topic (c) search-simulation 30-topic

Figure 11: SSP's performance and the number of iterations. The plots are the averages over: (a) all 30 data sets in the controlled-distribution collection, (b) ten 10-topic sets, and (c) ten 30-topic sets of the search-simulation data.

The dotted lines show the performance of the best-performing baselines for comparison. We see that the performance steeply improves in the first few iterations and becomes stable after two to four. This means that only a few iterations are necessary, and also excessive iterations would not hurt the performance. These are good characteristics.

5 Conclusion

We propose a new document clustering method SSP, which is driven by descriptor phrases. Our evaluation against human judgments shows that phrases contribute significantly to document clustering only when either IRR is used for phrase-document relation measure or when phrase clustering was conducted, subsumed under the document clustering. Our finding is that in order to exploit phrases in document clustering, methods need to have some mechanism for adequately representing the intrinsic relationships among phrases.

SSP outperforms a previously proposed IRR document vector-based clustering on all data types. Thus, we feel that exploring strategies for effective use of phrases is a promising direction towards further improvement of topical document clustering techniques.

Finally, the side products of phrase-driven clustering are cluster descriptor phrases and their trees (as shown in Figure 1). We have observed that they are useful as cluster summaries when presented to users. This warrants further investigation.

References

Rie Kubota Ando and Lillian Lee. 2001. Iterative Residual Rescaling: An analysis and generalization of LSI. In *Proceedings of SIGIR'01*, pages 154–162.

Branimir Boguraev and Mary Neff. 2000. Discourse segmentation in aid of document summarization. In *Proceedings of Hawaii International Conference on System Sciences (*HICSS-33*), Minitrack on Digital Documents Understanding*, Maui, Hawaii. IEEE.

Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. 1992. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proceedings of SIGIR'92*, pages 318–329.

Scott Deerwester, Susan T. Dumais, Geroge W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by Latent Semantic Analysis. *Journal of the Society for Information Science*, 41:391–407.

Ran El-Yaniv and Oren Souroujon. 2001. Iterative double clustering for unsupervised and semi-supervised learning. In *Proceedings of ECML-01, 12th European Conference on Machine Learning*, pages 121–132.

Martin Franz, J. Scott McCarley, Todd Ward, and Wei-Jing Zhu. 2001. Unsupervised and supervised clustering for topic tracking. In *Proceedings of SIGIR'01*, pages 310 – 317.

Vasileios Hatzivassiloglou, Luis Gravano, and Ankineedu Maganti. 2000. An investigation of linguistic features and clustering algorithms for topical document clustering. In *Proceedings of SIGIR'00*, pages 224 – 231.

Marti A. Hearst and Jan O. Pedersen. 1996. Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. In *Proceedings of SIGIR'96*.

Anton V. Leouski and W. Bruce Croft. 1996. An evaluation of techniques for clustering search results. In *Technical Report IR-76, Department of Computer Science, University of Massachusetts, Amherst*.

David D. Lewis. 1992. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of SIGIR'92*, pages 37–50.

Y. S. Maarek and A. J. Wecker. 1994. The librarian's assistant: automatically organizing on-line books into dynamic bookshelves. In *Proceedings of RIAO'94*.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, and London, England.

Gerald Salton and M.J. McGill. 1983. Introduction to modern information retrieval.

Hinrich Schütze and Craig Silverstein. 1997. Projections for efficient document clustering. In *Proceedings of SIGIR'97*, pages 74–81.

Craig Silverstein and Jan O. Pedersen. 1997. Almost-constant-time clustering of arbitrary corpus subsets. In *Proceedings of SIGIR'97*, pages 60–66.

Noam Slonim and Naftali Tishby. 2000. Document clustering using word clusters via the information bottleneck method. In *Proceedings of SIGIR'00*, pages 208–215.

Noam Slonim, Nir Friedman, and Naftali Tishby. 2002. Unsupervised document classification using sequential information maximization. In *Proceedings of SIGIR'02*, pages 129–136.

Michael B. W. Wolfe, M. E. Schreiner, Bob Rehder, and Darrell Laham. 1998. Learning from text: Matching readers and text by Latent Semantic Analysis. *Discourse Processes*, 25:309–336.

Jinxi Xu and W. Bruce Croft. 1999. Cluster-based language models for distributed retrieval. In *Proceedings of SIGIR'99*, pages 254 – 261.

Oren Zamir and Oren Etzioni. 1998. Web document clustering: A feasibility demonstration. In *Proceedings of SIGIR'98*.