Language Model-Based Document Clustering Using Random Walks

Güneş Erkan
Department of EECS
University of Michigan
Ann Arbor, MI 48109-2121
gerkan@umich.edu

Abstract

We propose a new document vector representation specifically designed for the document clustering task. Instead of the traditional term-based vectors, a document is represented as an n-dimensional vector, where n is the number of documents in the cluster. The value at each dimension of the vector is closely related to the generation probability based on the language model of the corresponding document. Inspired by the recent graph-based NLP methods, we reinforce the generation probabilities by iterating random walks on the underlying graph representation. Experiments with k-means and hierarchical clustering algorithms show significant improvements over the alternative $tf \cdot idf$ vector representation.

1 Introduction

Document clustering is one of the oldest and most studied problems of information retrieval (van Rijsbergen, 1979). Almost all document clustering approaches to date have represented documents as vectors in a *bag-of-words vector space model*, where each dimension of a document vector corresponds to a term in the corpus (Salton and McGill, 1983). General clustering algorithms are then applied to these vectors to cluster the given corpus. There have been attempts to use bigrams or even higher-order n-grams to represent documents in text categorization, the supervised counterpart of document clustering, with little success (Caropreso et al., 2001; Tan et al., 2002).

Clustering can be viewed as partitioning a set of data objects into groups such that the similarities between the objects in a same group is high while inter-group similarities are weaker. The fundamental assumption in this work is that *the documents that are likely to have been generated from similar language models are likely to be in the same cluster*. Under this assumption, we propose a new representation for document vectors specifically designed for clustering purposes.

Given a corpus, we are interested in the generation probabilities of a document based on the language models induced by other documents in the corpus. Using these probabilities, we propose a vector representation where each dimension of a document vector corresponds to a document in the corpus instead of a term in the classical representation. In other words, our document vectors are n-dimensional, where n is the number of documents in the corpus to be clustered. For the vector $\mathbf{v}_{d_i}$ of document d_i, the j^{th} element of $\mathbf{v_{d_i}}$ is closely related to the generation probability of d_i based on the language model induced by document d_j. The main steps of our method are as follows:

- For each ordered document pair (d_i, d_j) in a given corpus, we compute the generation probability of d_i from the language model induced by d_j making use of language-model approaches in information retrieval (Ponte and Croft, 1998).
- We represent each document by a vector of its generation probabilities based on other documents' language models. At this point, these vectors can be used in any clustering algorithm instead of the traditional term-based document vectors.
- Following (Kurland and Lee, 2005), our new document vectors are used to construct the underlying *generation graph*; the directed graph where documents are the nodes and link weights are proportional to the generation probabilities.
- We use *restricted random walk* probabilities to reinforce the generation probabilities and discover hidden relationships in the graph that are not obvious by the generation links. Our random walk model is similar to the one proposed by Harel and Kohen

Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 479–486,
New York, June 2006.

(2001) for general spatial data represented as undirected graphs. We have extended their model to the directed graph case. We use new probabilities derived from random walks as the vector representation of the documents.

2 Generation Probabilities as Document Vectors

2.1 Language Models

The language modeling approach to information retrieval was first introduced by Ponte and Croft (1998) as an alternative (or an improvement) to the traditional $tf \cdot idf$ relevance models. In the language modeling framework, each document in the database defines a language model. The relevance of a document to a given query is ranked according to the generation probability of the query based on the underlying language model of the document. To induce a (unigram) language model from a document, we start with the maximum likelihood (ML) estimation of the term probabilities. For each term w that occurs in a document D, the ML estimation of w with respect to D is defined as

$$p_{ML}(w|D) = \frac{\mathrm{tf}(w, D)}{\sum_{w' \in D} \mathrm{tf}(w', D)}$$

where $\mathrm{tf}(w, D)$ is the number of occurences of term w in document D. This estimation is often smoothed based on the following general formula:

$$p(w|D) = \lambda p_{ML}(w|D) + (1 - \lambda) p_{ML}(w|Corpus)$$

where $p_{ML}(w|Corpus)$ is the ML estimation of w over an entire corpus which usually D is a member of. λ is the general smoothing parameter that takes different forms in various smoothing methods. Smoothing has two important roles (Zhai and Lafferty, 2004). First, it accounts for terms unseen in the document preventing zero probabilities. This is similar to the smoothing effect in NLP problems such as parsing. Second, smoothing has an idf-like effect that accounts for the generation probabilities of the common terms in the corpus. A common smoothing technique is to use Bayesian smoothing with the Dirichlet prior (Zhai and Lafferty, 2004; Liu and Croft, 2004):

$$\lambda = \frac{\sum_{w' \in D} \mathrm{tf}(w', D)}{\sum_{w' \in D} \mathrm{tf}(w', D) + \mu}$$

Here, μ is the smoothing parameter. Higher values of μ mean more aggressive smoothing.

Assuming the terms in a text are independent from each other, the generation probability of a text sequence S given the document D is the product of the generation probabilities of the terms of S:

$$p(S|D) = \prod_{w \in S} p(w|D) \tag{1}$$

In the context of information retrieval, S is a query usually composed of few terms. In this work, we are interested in the generation probabilities of entire documents that usually have in the order of hundreds of unique terms. If we use Equation 1, we end up having unnatural probabilities which are irrepresentably small and cause floating point underflow. More importantly, longer documents tend to have much smaller generation probabilities no matter how closely related they are to the generating language model. However, as we are interested in the generation probabilities between all pairs of documents, we want to be able to compare two different generation probabilities from a fixed language model regardless of the target document sizes. This is not a problem in the classical document retrieval setting since the given query is fixed, and generation probabilities for different queries are not compared against each other. To address these problems, following (Lavrenko et al., 2002; Kurland and Lee, 2005), we "flatten" the probabilities by normalizing them with respect to the document size:

$$p_{\text{flat}}(S|D) = p(S|D)^{\frac{1}{|S|}} \tag{2}$$

where $|S|$ is the number of terms in S. p_{flat} provides us with meaningful values which are comparable among documents of different sizes.

2.2 Using Generation Probabilities as Document Representations

Equation 2 suggests a representation of the relationship of a document with the other documents in a corpus. Given a corpus of n documents to cluster, we form an n-dimensional *generation vector* $\mathbf{g}_{d_i} = (g_{d_i 1}, g_{d_i 2}, \ldots, g_{d_i n})$ for each document d_i where

$$g_{d_i j} = \begin{cases} 0 & \text{if } i = j, \\ p_{\text{flat}}(d_i|d_j) & \text{otherwise} \end{cases} \tag{3}$$

We can use these generation vectors in any clustering algorithm we prefer instead of the classical term-based $tf \cdot idf$ vectors. The intuition behind this idea becomes clearer when we consider the underlying directed graph representation, where each document is a node and the weight of the link from d_i to d_j is equal to $p_{\text{flat}}(d_i|d_j)$. An appropriate analogy here is the citation graph of scientific papers. The generation graph can be viewed as a model where documents *cite* each other. However, unlike real citations, the generation links are weighted and automatically induced from the content.

The similarity function used in a clustering algorithm over the generation vectors becomes a measure of structural similarity of two nodes in the generation graph. Work on bibliometrics uses various similarity metrics to assess the relatedness of scientific papers by looking at the citation vectors (Boyack et al., 2005). Graph-based

similarity metrics are also used to detect semantic similarity of two documents on the Web (Maguitman et al., 2005). Cosine, also the standard metric used in $tf \cdot idf$ based document clustering, is one of these metrics. Intuitively, the cosine of the citation vectors (i.e. vector of outgoing link weights) of two nodes is high when they link to similar sets of nodes with similar link weights. Hence, the cosine of two generation vectors is a measure of how likely two documents are generated from the same documents' language models.

The generation probability in Equation 2 with a smoothed language model is never zero. This creates two potential problems if we want to use the vector of Equation 3 directly in a clustering algorithm. First, we only want strong generation links to contribute in the similarity function since a low generation probability is not an evidence for semantic relatedness. This intuition is similar to throwing out the stopwords from the documents before constructing the $tf \cdot idf$ vectors to avoid coincidental similarities between documents. Second, having a dense vector with lots of non-zero elements will cause efficiency problems. Vector length is assumed to be a constant factor in analyzing the complexity of the clustering algorithms. However, our generation vectors are n-dimensional, where n is the number of documents. In other words, vector size is not a constant factor anymore, which causes a problem of scalability to large data sets. To address these problems, we use what Kurland and Lee (2005) define as *top generators*: Given a document d_i, we consider only c documents that yield the largest generation probabilities and discard others. The resultant n-dimensional vector, denoted $\mathbf{g}^c_{d_i}$, has at most c non-zero elements, which are the largest c elements of $\mathbf{g}_{d_i}$. For a given constant c, with a sparse vector representation, certain operations (e.g. cosine) on such vectors can be done in constant time independent of n.

2.3 Reinforcing Links with Random Walks

Generation probabilities are only an approximation of semantic relatedness. Using the underlying directed graph interpretation of the generation probabilities, we aim to get better approximations by accumulating the generation link information in the graph. We start with some definitions. We denote a (directed) graph as $G(V, w)$ where V is the set of nodes and $w : V \times V \rightarrow \mathbb{R}$ is the *link weight function*. We formally define a generation graph as follows:

Definition 1 *Given a corpus $\mathcal{C} = \{d_1, d_2, \ldots, d_n\}$ with n documents, and a constant c, the* generation graph *of $\mathcal{C}$ is a directed graph $G_c(\mathcal{C}, w)$, where $w(d_i, d_j) = g^c_{d_i j}$.*

Definition 2 *A t-step* random walk *on a graph $G(V, w)$ that starts at node $v_0 \in V$ is a sequence of nodes $v_0, v_1, \ldots, v_t \in V$ where $w(v_i, v_{i+1}) > 0$ for all $0 \leq i < t$. The* probability *of a t-step random walk is defined as $\prod_{i=0}^{t-1} q_{v_i v_{i+1}}$ where*

$$q_{v_i v_{i+1}} = \frac{w(v_i, v_{i+1})}{\sum_{u \in V} w(v_i, u)}$$

q_{uv} is called the transition probability *from node u to node v.*

For example, for a generation graph G_c, there are at most c 1-step random walks that start at a given node with probabilities proportional to the weights of the outgoing generation links of that node.

Suppose there are three documents A, B, and C in a generation graph. Suppose also that there are "strong" generation links from A to B and B to C, but no link from A to C. The intuition says that A must be semantically related to C to a certain degree although there is no generation link between them depending on C's language model. We approximate this relation by considering the probabilities of 2-step (or longer) random walks from A to C although there is no 1-step random walk from A to C.

Let q^t_{uv} denote the probability that an t-step random walk starts at u and ends at v. An interesting property of random walks is that for a given node v, q^∞_{uv} does not depend on u. In other words, the probability of a random walk ending up at v "in the long run" does not depend on its starting point (Seneta, 1981). This limiting probability distribution of an infinite random walk over the nodes is called the *stationary distribution* of the graph. The stationary distribution is uninteresting to us for clustering purposes since it gives an information related to the global structure of the graph. It is often used as a measure to *rank* the structural importance of the nodes in a graph (Brin and Page, 1998). For clustering, we are more interested in the local similarities inside a "cluster" of nodes that separate them from the rest of the graph. Furthermore, the generation probabilities lose their significance during long random walks since they get multiplied at each step. Therefore, we compute q^t for small values of t. Finally, we define the following:

Definition 3 *The t-*step generation probability *of document d_i from the language model of d_j:*

$$gen^t(d_i|d_j) = \frac{\sum_{s=1}^{t} q^s_{d_i d_j}}{t}$$

*$\mathbf{gen}^t_{d_i} = (gen^t(d_i|d_1), gen^t(d_i|d_2), \ldots, gen^t(d_i|d_n))$ is the t-*step generation vector *of document d_i. We will often write $\mathbf{gen}^t$ omitting the document name when we are not talking about the vector of a specific document.*

$\text{gen}_t(d_i, d_j)$ is a measure of how likely a random walk that starts at d_i will visit d_j in t or fewer steps. It helps us to discover "hidden" similarities between documents

that are not immediately obvious from 1-step generation links. Note that when $t = 1$, $\mathbf{gen}^1_{d_i}$ is nothing but $\mathbf{g}^c_{d_i}$ normalized such that the sum of the elements of the vector is 1. The two are practically the same representations since we compute the cosine of the vectors during clustering.

3 Related Work

Our work is inspired by three main areas of research. First, the success of language modeling approaches to information retrieval (Ponte and Croft, 1998) is encouraging for a similar twist to document representation for clustering purposes. Second, graph-based inference techniques to discover "hidden" textual relationships like the one we explored in our random walk model have been successfully applied to other NLP problems such as summarization (Erkan and Radev, 2004; Mihalcea and Tarau, 2004; Zha, 2002), prepositional phrase attachment (Toutanova et al., 2004), and word sense disambiguation (Mihalcea, 2005). Unlike our approach, these methods try to exploit the global structure of a graph to *rank* the nodes of the graph. For example, Erkan and Radev (2004) find the stationary distribution of the random walk on a graph of sentences to rank the salience scores of the sentences for extractive summarization. Their link weight function is based on cosine similarity. Our graph construction based on generation probabilities is inherited from (Kurland and Lee, 2005), where authors used a similar generation graph to rerank the documents returned by a retrieval system based on the stationary distribution of the graph. Finally, previous research on clustering graphs with restricted random walks inspired us to cluster the generation graph using a similar approach. Our t-step random walk approach is similar to the one proposed by Harel and Koren (2001). However, their algorithm is proposed for "spatial data" where the nodes of the graph are connected by undirected links that are determined by a (symmetric) similarity function. Our contribution in this paper is to use their approach on textual data by using generation links, and extend the method to directed graphs.

There is an extensive amount of research on document clustering or clustering algorithms in general that we can not possibly review here. After all, we do not present a new clustering algorithm, but rather a new representation of textual data. We explain some popular clustering algorithms and evaluate our representation using them in Section 4. Few methods have been proposed to cluster documents using a representation other than the traditional $tf \cdot idf$ vector space (or similar term-based vectors). Using a bipartite graph of terms and documents and then clustering this graph based on spectral methods is one of them (Dhillon, 2001; Zha et al., 2001). There are also general spectral methods that start with $tf \cdot idf$ vectors, then map them to a new space with fewer dimensions before initiating the clustering algorithm (Ng et al., 2001).

The *information-theoretic* clustering algorithms are relevant to our framework in the sense that they involve probability distributions over words just like the language models. However, instead of looking at the word distributions at the individual document level, they make use of the *joint* distribution of words and documents. For example, given the set of documents X and the set of words Y in the document collection, Slonim and Tishby (2000) first try to find a word clustering $\hat{Y}$ such that the mutual information $I(Y, \hat{Y})$ is minimized (for good compression) while maximizing the $I(\hat{Y}, X)$ (for preserving the original information). Then the same procedure is used for clustering documents using the word clusters from the first step. Dhillon et. al. (2003) propose a *co-clustering* version of this information-theoretic method where they cluster the words and the documents concurrently.

4 Evaluation

We evaluated our new vector representation by comparing it against the traditional $tf \cdot idf$ vector space representation. We ran k-means, single-link, average-link, and complete-link clustering algorithms on various data sets using both representations. These algorithms are among the most popular ones that are used in document clustering.

4.1 General Experimental Setting

Given a corpus, we stemmed all the documents, removed the stopwords and constructed the $tf \cdot idf$ vector for each document by using the *bow toolkit* (McCallum, 1996). We computed the idf of each term using the following formula:

$$\text{idf}(w) = log_2 \left(\frac{n}{\text{df}(w)} \right)$$

where n is the total number of documents and $\text{df}(w)$ is the number of documents that the term w appears in.

We computed flattened generation probabilities (Equation 2) for all ordered pairs of documents in a corpus, and then constructed the corresponding generation graph (Definition 1). We used Dirichlet-smoothed language models with the smoothing parameter $\mu = 1000$, which can be considered as a typical value used in information retrieval. While computing the generation link vectors, we did not perform extensive parameter tuning at any stage of our method. However, we observed the following:

- When c (number of outgoing links per document) was very small (less than 10), our methods performed poorly. This is expected with such a sparse vector representation for documents. However, the performance got rapidly and almost monotonically

better as we increased c until around $c = 80$, where the performance stabilized and dropped after around $c = 100$. We conclude that using bounded number of outgoing links per document is not only more efficient but also necessary as we motivated in Section 2.2.

- We got the best results when the random walk parameter $t = 3$. When $t > 3$, the random walk goes "out of the cluster" and $\mathbf{gen}^t$ vectors become very dense. In other words, almost all of the graph is reachable from a given node with 4-step or longer random walks (assuming c is around 80), which is an indication of a "small world" effect in generation graphs (Watts and Strogatz, 1998).

Under these observations, we will only report results using vectors $\mathbf{gen}^1$, $\mathbf{gen}^2$ and $\mathbf{gen}^3$ with $c = 80$ regardless of the data set and the clustering algorithm.

4.2 Experiments with k-means

4.2.1 Algorithm

k-means is a clustering algorithm popular for its simplicity and efficiency. It requires k, the number of clusters, as input, and partitions the data set into exactly k clusters. We used a version of k-means that uses cosine similarity to compute the distance between the vectors. The algorithm can be summarized as follows:

1. randomly select k document vectors as the initial cluster centroids;
2. assign each document to the cluster whose centroid yields the highest cosine similarity;
3. recompute the centroid of each cluster. (centroid vector of a cluster is the average of the vectors in that cluster);
4. stop if none of the centroid vectors has changed at step 3. otherwise go to step 2.

4.2.2 Data

k-means is known to work better on data sets in which the documents are nearly evenly distributed among different clusters. For this reason, we tried to pick such corpora for this experiment to be able to get a fair comparison between different document representations. The first corpus we used is *classic3*,[1] which is a collection of technical paper abstracts in three different areas. We used two corpora, *bbc* and *bbcsport*, that are composed of BBC news articles in general and sports news, respectively. [2] Both corpora have 5 news classes each. *20news*[3] is a corpus of newsgroup articles composed of 20 classes. Table 1 summarizes the corpora we used together with the sizes of the smallest and largest class in each of them.

Corpus	Documents	Classes	Smallest	Largest
classic3	3891	3	1033	1460
bbcsport	737	5	100	265
bbc	2225	5	386	511
20news	18846	20	628	999

Table 1: The corpora used in the k-means experiments.

4.2.3 Results

We used two different metrics to evaluate the results of the k-means algorithm; accuracy and mutual information. Let l_i be the label assigned to d_i by the clustering algorithm, and α_i be d_i's actual label in the corpus. Then,

$$\text{Accuracy} = \frac{\sum_{i=1}^{n} \delta(\text{map}(l_i), \alpha_i)}{n}$$

where $\delta(x, y)$ equals 1 if $x = y$ and equals zero otherwise. $\text{map}(l_i)$ is the function that maps the output label set of the k-means algorithm to the actual label set of the corpus. Given the confusion matrix of the output, best such mapping function can be efficiently found by Munkres's algorithm (Munkres, 1957).

Mutual information is a metric that does not require a mapping function. Let $L = \{l_1, l_2, \ldots, l_k\}$ be the output label set of the k-means algorithm, and $A = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ be the actual label set of the corpus with the underlying assignments of documents to these sets. Mutual information (MI) of these two labelings is defined as:

$$\text{MI}(L, A) = \sum_{l_i \in L, \alpha_j \in A} P(l_i, \alpha_j) \cdot \log_2 \frac{P(l_i, \alpha_j)}{P(l_i) \cdot P(\alpha_j)}$$

where $P(l_i)$ and $P(\alpha_j)$ are the probabilities that a document is labeled as l_i and α_j by the algorithm and in the actual corpus, respectively; $P(l_i, \alpha_j)$ is the probability that these two events occur at the same time. These values can be derived from the confusion matrix. We map the MI metric to the $[0, 1]$ interval by normalizing it with the maximum possible MI that can be achieved with the corpus. Normalized MI is defined as

$$\widehat{\text{MI}} = \frac{\text{MI}(L, A)}{\text{MI}(A, A)}$$

[1] `ftp://ftp.cs.cornell.edu/pub/smart`

[2] `http://www.cs.tcd.ie/Derek.Greene/research/datasets.html` BBC corpora came in preprocessed format so that we did not perform the processing with the *bow* toolkit mentioned in Section 4.1

[3] `http://people.csail.mit.edu/jrennie/20Newsgroups`

One disadvantage of k-means is that its performance is very dependent on the initial selection of cluster centroids. Two approaches are usually used when reporting the performance of k-means. The algorithm is run multiple times; then either the average performance of these runs or the best performance achieved is reported. Reporting the best performance is not very realistic since we would not be clustering a corpus if we already knew the class labels. Reporting the average may not be very informative since the variance of multiple runs is usually large. We adopt an approach that is somewhere in between. We use "true seeds" to initialize k-means, that is, we *randomly* select k document vectors *that belong to each of the true classes* as the initial centroids. This is not an unrealistic assumption since we initially know the number of classes, k, in the corpus, and the cost of finding one example document from each class is not usually high. This way, we also aim to reduce the variance of the performance of different runs for a better analysis.

Table 2 shows the results of k-means algorithm using $tf \cdot idf$ vectors versus generation vectors $\mathbf{gen}^1$ (plain flattened generation probabilities), $\mathbf{gen}^2$ (2-step random walks), $\mathbf{gen}^3$ (3-step random walks). Taking advantage of the relatively larger size and number of classes of *20news* corpus, we randomly divided it into disjoint partitions with 4, 5, and 10 classes which provided us with 5, 4, and 2 new corpora, respectively. We named them *4news-1*, *4news-2*, ..., *10news-2* for clarity. We ran k-means with 30 distinct initial seed sets for each corpus.

The first observation we draw from Table 2 is that even $\mathbf{gen}^1$ vectors perform better than the $tfidf$ model. This is particularly surprising given that $\mathbf{gen}^1$ vectors are sparser than the $tf \cdot idf$ representation for most documents.[4] All $\mathbf{gen}^t$ vectors clearly outperform $tf \cdot idf$ model often by a wide margin. The performance also gets better (not always significantly though) in almost all data sets as we increase the random walk length, which indicates that random walks are useful in reinforcing generation links and inducing new relationships. Another interesting observation is that the confidence intervals are also narrower for generation vectors, and tend to get even narrower as we increase t.

4.3 Experiments with Hierarchical Clustering

4.3.1 Algorithms

Hierarchical clustering algorithms start with the trivial clustering of the corpus where each document defines a separate cluster by itself. At each iteration, two "most similar" separate clusters are merged. The algorithm stops after $n-1$ iterations when all the documents are merged into a single cluster.

Hierarchical clustering algorithms differ in how they define the similarity between two clusters at each merging step. We experimented with three of the most popular algorithms using cosine as the similarity metric between two vectors. *Single-link clustering* merges two clusters whose most similar members have the highest similarity. *Complete-link clustering* merges two clusters whose least similar members have the highest similarity. *Average-link clustering* merges two clusters that yield the highest average similarity between all pairs of documents.

4.3.2 Data

Corpus	Documents	Classes	Smallest	Largest
Reuters	8646	57	2	3735
TDT2	10160	87	2	1843

Table 3: The corpora used in the hierarchical clustering experiments.

Although hierarchical algorithms are not very efficient, they are useful when the documents are not evenly distributed among the classes in the corpus and some classes exhibit a "hierarchical" nature; that is, some classes in the data might be semantically overlapping or they might be in a subset/superset relation with each other. We picked two corpora that may exhibit such nature to a certain extent. Reuters-21578[5] is a collection of news articles from Reuters. TDT2[6] is a similar corpus of news articles collected from six news agencies in 1998. They contain documents labeled with zero, one or more class labels. For each corpus, we used only the documents with exactly one label. We also eliminated classes with only one document since clustering such classes is trivial. We ended up with two collections summarized in Table 3.

4.3.3 Results

The output of a hierarchical clustering algorithm is a *tree* where leaves are the documents and each node in the tree shows a cluster merging operation. Therefore each subtree represents a cluster. We assume that each class of documents in the corpus form a cluster subtree at some point during the construction of the tree. To evaluate the cluster tree, we use F-measure proposed in (Larsen and Aone, 1999). F-measure for a class c_i in the corpus and a subtree s_j is defined as

$$F(c_i, s_j) = \frac{2 \cdot R(c_i, s_j) \cdot P(c_i, s_j)}{R(c_i, s_j) + P(c_i, s_j)}$$

[4] Remember that we set $c = 80$ in our experiments which means that there can be a maximum of 80 non-zero elements in $\mathbf{gen}^1$. Most documents have more than 80 unique terms in them.

[5] http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

[6] http://www.nist.gov/speech/tests/tdt/tdt98/index.htm

Corpus	k	Accuracy ($\times 100$)				Normalized Mutual Information ($\times 100$)			
		$tf \cdot idf$	$\mathbf{gen}^1$	$\mathbf{gen}^2$	$\mathbf{gen}^3$	$tf \cdot idf$	$\mathbf{gen}^1$	$\mathbf{gen}^2$	$\mathbf{gen}^3$
classic3	3	95.76 ± 1.28	97.92 ± 0.69	98.70 ± 0.19	$\mathbf{98.79 \pm 0.03}$	84.69 ± 2.50	91.16 ± 1.90	93.39 ± 0.59	$\mathbf{93.64 \pm 0.12}$
4news-1	4	72.32 ± 2.80	82.76 ± 1.55	85.86 ± 1.30	$\mathbf{86.58 \pm 1.23}$	49.85 ± 3.39	64.85 ± 1.56	69.72 ± 0.88	$\mathbf{70.73 \pm 0.89}$
4news-2	4	63.42 ± 2.83	77.33 ± 2.14	80.66 ± 1.92	$\mathbf{81.29 \pm 1.80}$	34.02 ± 2.97	54.55 ± 1.90	59.50 ± 1.45	$\mathbf{60.46 \pm 1.31}$
4news-3	4	62.35 ± 2.51	74.96 ± 3.35	78.60 ± 3.67	$\mathbf{78.85 \pm 3.75}$	33.74 ± 2.65	51.94 ± 3.43	58.15 ± 3.08	$\mathbf{59.24 \pm 2.95}$
4news-4	4	73.14 ± 2.05	81.36 ± 1.54	83.54 ± 1.50	$\mathbf{84.22 \pm 1.49}$	54.24 ± 2.74	66.47 ± 1.24	69.78 ± 0.90	$\mathbf{70.41 \pm 0.83}$
4news-5	4	69.43 ± 3.68	87.06 ± 2.33	$\mathbf{90.07 \pm 1.66}$	90.06 ± 1.47	42.58 ± 3.90	66.49 ± 3.27	71.95 ± 2.45	$\mathbf{71.96 \pm 2.15}$
5news-1	5	59.00 ± 2.43	73.91 ± 2.30	76.35 ± 2.86	$\mathbf{76.75 \pm 2.58}$	36.53 ± 2.94	56.84 ± 3.19	60.74 ± 2.87	$\mathbf{61.40 \pm 2.54}$
5news-2	5	55.59 ± 2.88	69.32 ± 2.17	72.75 ± 1.74	$\mathbf{73.05 \pm 1.82}$	30.94 ± 2.45	48.00 ± 2.18	52.77 ± 1.52	$\mathbf{53.75 \pm 1.30}$
5news-3	5	71.07 ± 2.39	83.78 ± 2.16	86.14 ± 2.15	$\mathbf{86.28 \pm 2.22}$	49.65 ± 2.64	67.09 ± 2.12	71.13 ± 1.87	$\mathbf{71.70 \pm 1.67}$
5news-4	5	70.06 ± 2.64	80.20 ± 1.84	82.61 ± 1.69	$\mathbf{82.66 \pm 1.78}$	50.04 ± 2.82	63.69 ± 1.73	66.89 ± 1.32	$\mathbf{67.49 \pm 1.14}$
bbc	5	80.73 ± 2.80	87.46 ± 2.63	89.88 ± 2.57	$\mathbf{90.80 \pm 2.26}$	63.34 ± 3.23	74.19 ± 2.93	78.20 ± 2.63	$\mathbf{79.39 \pm 2.30}$
bbcsport	5	79.25 ± 2.87	94.25 ± 0.89	95.44 ± 0.42	$\mathbf{95.56 \pm 0.31}$	63.94 ± 3.27	84.59 ± 1.34	86.42 ± 0.71	$\mathbf{86.54 \pm 0.58}$
10news-1	10	50.11 ± 2.30	66.20 ± 2.12	69.18 ± 1.73	$\mathbf{69.76 \pm 1.61}$	39.98 ± 1.99	55.21 ± 1.67	58.86 ± 1.15	$\mathbf{59.70 \pm 0.96}$
10news-2	10	54.12 ± 1.76	70.01 ± 2.00	73.41 ± 1.78	$\mathbf{74.14 \pm 1.75}$	42.44 ± 1.55	60.64 ± 1.40	65.20 ± 1.11	$\mathbf{66.41 \pm 1.02}$
20news	20	41.46 ± 1.03	55.75 ± 1.25	58.97 ± 1.29	$\mathbf{60.26 \pm 0.99}$	38.28 ± 0.73	52.44 ± 0.74	56.34 ± 0.71	$\mathbf{57.47 \pm 0.53}$

Table 2: Performances of different vector representations using k-means (average of 30 runs $\pm 95\%$ confidence interval).

Algorithm	TDT2				Reuters-21578			
	$tf \cdot idf$	$\mathbf{gen}^1$	$\mathbf{gen}^2$	$\mathbf{gen}^3$	$tf \cdot idf$	$\mathbf{gen}^1$	$\mathbf{gen}^2$	$\mathbf{gen}^3$
single-link	65.25	82.96	**84.22**	83.92	59.35	59.37	65.70	**66.15**
average-link	90.78	93.53	94.04	**94.13**	78.25	79.17	77.24	**81.37**
complete-link	29.07	25.04	27.19	**34.67**	43.66	42.79	45.91	**48.36**

Table 4: Performances (F-measure $\times 100$) of different vector representations using hierarchical algorithms on two corpora.

where $R(c_i, s_j)$ and $P(c_i, s_j)$ is the recall and the precision of s_j considering the class c_i. Let S be the set of subtrees in the output cluster tree, and C be the set of classes. F-measure of the entire tree is the weighted average of the maximum F-measures of all the classes:

$$F(C, S) = \sum_{c \in C} \frac{n_c}{n} \max_{s \in S} F(c, s)$$

where n_c is the number of documents that belong to class c.

We ran all three algorithms for both corpora. Unlike k-means, hierarchical algorithms we used are deterministic. Table 4 summarizes our results. An immediate observation is that average-link clustering performs much better than other two algorithms independent of the data set or the document representation, which is consistent with earlier research (Zhao and Karypis, 2002). The highest result (shown boldface) for each algorithm and corpus was achieved by using generation vectors. However, unlike in the k-means experiments, $tf \cdot idf$ was able to outperform $\mathbf{gen}^1$ and $\mathbf{gen}^2$ in one or two cases. $\mathbf{gen}^2$ yielded the best result instead of $\mathbf{gen}^3$ in one of the six cases.

5 Conclusion

We have presented a language model inspired approach to document clustering. Our results show that even the simplest version of our approach with nearly no parameter tuning can outperform traditional $tf \cdot idf$ models by a wide margin. Random walk iterations on our graph-based model have improved our results even more. Based on the success of our model, we will investigate various graph-based relationships for explaining semantic structure of text collections in the future. Possible applications include information retrieval, text clustering/classification and summarization.

Acknowledgments

I would like to thank Dragomir Radev for his useful comments. This work was partially supported by the U.S. National Science Foundation under the following two grants: 0329043 "Probabilistic and link-based Methods for Exploiting Very Large Textual Repositories" administered through the IDM program and 0308024 "Collaborative Research: Semantic Entity and Relation Extraction from Web-Scale Text Document Collections" administered by the HLT program. All opinions, findings, conclusions, and recommendations in this paper are made by the authors and do not necessarily reflect the views of the National Science Foundation.

References

Kevin W. Boyack, Richard Klavans, and Katy Börner. 2005. Mapping the backbone of science. *Scientometrics*, 64(3):351–374.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the*

7th International World Wide Web Conference, pages 107–117.

Maria Fernanda Caropreso, Stan Matwin, and Fabrizio Sebastiani. 2001. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In Amita G. Chin, editor, *Text Databases and Document Management: Theory and Practice*, pages 78–102. Idea Group Publishing, Hershey, US.

Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. 2003. Information-theoretic co-clustering. In Pedro Domingos, Christos Faloutsos, Ted SEnator, Hillol Kargupta, and Lise Getoor, editors, *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-03)*, pages 89–98, New York, August 24–27. ACM Press.

Inderjit S. Dhillon. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the Seventh ACM SIGKDD Conference*, pages 269–274.

Güneş Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

David Harel and Yehuda Koren. 2001. Clustering spatial data using random walks. In *Proceedings of the Seventh ACM SIGKDD Conference*, pages 281–286, New York, NY, USA. ACM Press.

Oren Kurland and Lillian Lee. 2005. PageRank without hyperlinks: Structural re-ranking using links induced by language models. In *Proceedings of SIGIR*.

Bjornar Larsen and Chinatsu Aone. 1999. Fast and effective text mining using linear-time document clustering. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22, New York, NY, USA. ACM Press.

Victor Lavrenko, James Allan, Edward DeGuzman, Daniel LaFlamme, Veera Pollard, and Stephen Thomas. 2002. Relevance models for topic detection and tracking. In *Proceedings of HLT*, pages 104–110.

Xiaoyong Liu and W. Bruce Croft. 2004. Cluster-based retrieval using language models. In *Proceedings of SIGIR*, pages 186–193.

Ana G. Maguitman, Filippo Menczer, Heather Roinestad, and Alessandro Vespignani. 2005. Algorithmic detection of semantic similarity. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 107–116, New York, NY, USA. ACM Press.

Andrew Kachites McCallum. 1996. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 404–411, Barcelona, Spain, July. Association for Computational Linguistics.

Rada Mihalcea. 2005. Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 411–418, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

James Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, March.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856.

Jay M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of SIGIR*, pages 275–281.

G. Salton and M. J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw Hill.

E. Seneta. 1981. *Non-negative matrices and markov chains*. Springer-Verlag, New York.

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

Chade-Meng Tan, Yuan-Fang Wang, and Chan-Do Lee. 2002. The use of bigrams to enhance text categorization. *Inf. Process. Manage*, 38(4):529–546.

Kristina Toutanova, Christopher D. Manning, and Andrew Y. Ng. 2004. Learning random walk models for inducing word dependency distributions. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 103, New York, NY, USA. ACM Press.

Cornelis J. van Rijsbergen. 1979. *Information Retrieval*. Butterworths.

Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, June 4.

Hongyuan Zha, Xiaofeng He, Chris H. Q. Ding, Ming Gu, and Horst D. Simon. 2001. Bipartite graph partitioning and data clustering. In *Proceedings of CIKM*, pages 25–32.

Hongyuan Zha. 2002. Generic Summarization and Key Phrase Extraction Using Mutual Reinforcement Principle and Sentence Clustering. Tampere, Finland.

Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst. (TOIS)*, 22(2):179–214.

Ying Zhao and George Karypis. 2002. Evaluation of hierarchical clustering algorithms for document datasets. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 515–524, New York, NY, USA. ACM Press.