USING GRAPHS AND RANDOM WALKS FOR DISCOVERING LATENT SEMANTIC RELATIONSHIPS IN TEXT

by

Güneş Erkan

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2007

Doctoral Committee:

Associate Professor Dragomir R. Radev, Chair
Associate Professor Steven P. Abney
Associate Professor Satinder Singh Baveja
Associate Professor Mark E. Newman
Assistant Professor Lada A. Adamic

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

USING GRAPHS AND RANDOM WALKS FOR DISCOVERING LATENT SEMANTIC RELATIONSHIPS IN TEXT

by
Güneş Erkan

Chair: Dragomir R. Radev

We propose a graph-based representation of text collections where the nodes are textual units such as sentences or documents, and the edges represent the pairwise similarity function between these units. We show how random walks on such a graph can give us better approximations for the latent similarities between two natural language strings. We also derive algorithms based on random walk models to rank the nodes in a text similarity graph to address the text summarization problem in information retrieval. The similarity functions used in the graphs are intentionally chosen to be very simple and language-independent to make our methods as generic as possible, and to show that significant improvements can be achieved even by starting with such similarity functions. We put special emphasis on language modeling-based similarity functions since we use them for the first time on problems such as document clustering and classification, and get improved results compared to the classical similarity functions such as cosine. Our graph-based methods are applicable to a diverse set of problems including generic and focused summarization, document clustering, and text classication. The text summarization system we have developed has ranked

as one of the top systems in Document Understanding Conferences over the past few years. In document clustering and classification, using language modeling functions performs consistently better than using the classical cosine measure reaching as high as 25% improvement in accuracy. Random walks on the similarity graph achieve additional significant improvements on top of this. We also revisit the nearest neighbor text classification methods and derive semi-supervised versions by using random walks that rival the state-of-the-art classification algorithms such as Suppor Vector Machines.

CHAPTER I

INTRODUCTION

Assessing the similarities among the strings of a language is key to solving most problems in Information Retrieval (IR) and Natural Language Processing (NLP) defined at different granularities (e.g. words, sentences, or documents) of text. In this work, we interpret *similarity* as a general abstract concept which is a measure of the amount of common information between two strings of a language. Throughout this thesis, we will be introducing different mathematical definitions of similarity depending on the specific problem we deal with. For example, the popular *cosine* measure in IR, which we introduce in Chapter II, looks at the number of words that two strings have in common normalized by the length of the strings.

Often there is an asymmetry between two strings in terms of the amount of information each one contains. Consider the following two sentences:

Example I.1.

S_1: *John loves Mary.*

S_2: *John loves Mary, but she does not know this.*

S_2 clearly contains all the information S_1 has and some more. A *symmetric* similarity function such as cosine cannot capture this phenomenon. Therefore, we will also look at asymmetric functions to measure the common information content in a

given pair of strings. For this purpose, we will adopt the concepts of *language models* and *generation probabilities* from the NLP literature.

Another property of natural language strings is the transitivity of the common or related information. Suppose we have three documents, D_1, D_2, and D_3. Knowing that D_1 is similar to D_2, and D_2 is similar to D_3, we can argue that D_1 and D_3 should be similar or related at least to a certain extent although they might not have many words in common. However, basic similarity measures such as *cosine* we have mentioned above depend on word overlap between two strings and often fail to capture this kind of *latent* similarities. As an example, consider these three short documents:

Example I.2.

D_1: *Michigan*

D_2: *Michigan Ann Arbor*

D_3: *Ann Arbor*

We know that *Michigan* and *Ann Arbor* are related, so D_1 and D_3 are somehow related. Nevertheless, it is hard to induce this relation with a measure based on word overlap between D_1 and D_3. A popular method in information retrieval is to use *relevance feedback* (Salton and Buckley, 1990) to solve the relatedness problem. For example, we might be interested in retrieving documents about Michigan and provide a retrieval system with query *Michigan*. Relevance feedback *transforms* the original query based on the initially retrieved documents and runs this new transformed query. For example, if *Ann Arbor* shows up a lot in the retrieved documents, we might want to include it in the next query as well. This way, more and more relevant documents are retrieved, and the recall of the retrieval system is improved.

There are several problems with this method. How many times we should transform the original query is not obvious. Expanding the query too much can take us to irrelevant documents. Also, transforming the original query is a challenging task. There are numerous thresholds and decisions based on these thresholds to determine which words to include or exclude in the new query. In this thesis, we address this problem of relevance or similarity *propagation* with a more principled and unified method such that ad-hoc parameters and domain-specific decisions are mostly eliminated. *This is where the contribution of our work comes into play.* We propose a graph-based representation of a collection of language strings where each node in the graph represents an individual string (e.g. a sentence or a document), and the edges are defined by a pairwise similarity relation between these strings. The edges may be directed or undirected, or weighted or unweighted depending on the similarity function we consider. The motivation behind this representation is that the popular pairwise similarity measures used in IR and NLP are usually poor approximations of the real content similarity, but we can induce better approximations only by reiterating these simple similarity measures without using any other external knowledge sources. For example, if we represent the documents in Example I.2 as a graph and use a simple similarity measure for the edge relation, no edge exists between D_1 and D_3. However, graph theory provides us with a number of tools to induce relations among the nodes that are *indirectly* connected to each other via paths that are longer than one link, such as D_1 and D_3. In general, given enough data, we claim that the relationship between semantically related strings can be inferred from the relationships in the graphs constructed by simple similarity functions. For instance, the sentences that have *Michigan* and/or *Ann Arbor* in them will be more strongly connected to each other either directly or by only a few edges in a similarity graph in

comparison to two randomly selected strings. The particular method we focus on in this thesis is using *random walks* on the similarity graphs of text. We will be using ideas from the theory of random walks and graphs in general to better approximate the underlying similarity relations between the strings.

Our methods proposed in this thesis do not use any external information other than the input text for a given problem. All the inferences about the relationships among the input strings are based on the basic similarity functions and the graph structure built on top of these functions specific to the problem we deal with. This enables us to test our hypotheses more directly and makes our results more conclusive. It also shows that our approach is potentially effective on a wide range of domains and languages where additional information sources are not easy to find. However, we want to point out that our methods do not necessarily exclude the use of external knowledge sources. For example, a lexical database such as WordNet (Fellbaum, 1998) could be used to retrieve synonyms and related words in English. This information could be incorporated into the similarity functions, and our methods could still be used on top of these improved functions. Therefore, such improvements based on already existing sources are orthogonal to our methods proposed in this thesis.

It is possible to think of the similarity concept in text without dealing with graphs. We can explain all of our methods we propose here without mentioning graphs at all. However, using graphs has a number of advantages. First of all, graphs enable us to visualize the data as a more compact and intuitive representation. Second, our methods can rest on well-established foundations that have underlain the study of graphs for years. Graph theory gives us tools that we can draw direct analogies from our methods and intuitions behind them. Strings become nodes, similarity relationships become edges. The relationships among nodes can be modeled in terms

of random walks on graphs.

Using graphs to model human language has become quite popular in recent years. Some of these efforts have focused on the analysis of human language itself rather than solving specific NLP and IR problems. One line of research uses graphs of individual words, also known as *lexical networks*, where the edges among the words are based on their cooccurence statistics. Such graphs have been used to build better probabilistic models of language (Jedynak and Karakos, 2007), or even to explain the organization and the evolution of human language (Ferrer i Cancho and Solé, 2001; Dorogovtsev and Mendes, 2001). Our purpose in this thesis is quite different and more practical. We aim to use graphs based on textual similarity to induce better similarity relationships among the language strings and hence to improve results on a number of NLP and IR problems. Due to the definitions of the specific problems we consider, we only use similarity graphs in which nodes are either sentences or documents. Other graph-based methods of similar nature and motivation have recently been proposed, which we cover in the related work sections of the subsequent chapters. Most of them have occured concurrently with the work we present in this thesis. Even yet some have been inspired by our work. For example, our graph-based approach to text summarization (Chapter III) has led to similar studies that analyze or extend it (Lin and Kan, 2007; Leite et al., 2007).

It is hard to evaluate the accuracy of a similarity or relatedness measure directly. There do not exist many text collections where the pairwise similarities of the text units have been assessed explicitly by humans. Furthermore, since similarity is a vague and subjective concept, its definition can change in the contexts of different problems. Therefore, we will evaluate our proposed methods indirectly by looking at some NLP problems where assessment of similarity is very important. Rather

than evaluating the similarity values directly, we will evaluate the improvements we get in several problems by replacing the classical similarity based approaches with our new graph-based similarity methods. This will also demonstrate that our ideas and graph-based methods in this thesis can be reused for effectively solving a diverse set of problems. The order of the chapters in this thesis follows our research on graph-based methods for NLP and IR chronologically, and reflects how the problems studied have motivated us to develop ideas for the subsequent ones.

Chapter III addresses the problem of *generic text summarization*, that is, producing a short summary of a given document or a set of documents such that the summary reflects as much information from the original documents as possible. We focus on *extractive* summarization which aims to select a few sentences verbatim from the original documents to produce a summary by concatenating them. Starting with the very early summarization systems, the intuition has been that a good summary should include the information that frequently occurs in the documents to be summarized. The simplest approach is to pick the sentences that contain the most frequent words in the set of documents (Luhn, 1958). Since this ignores the relationships among the words, several methods have been proposed to improve it. One influencial method is to look at *sequences* of words, called *lexical chains* (Morris and Hirst, 1991; Barzilay and Elhadad, 1997), instead of individual words to determine strongly related words around a topic and select the summary sentences accordingly.

Our approach takes this one step further by looking directly at the relationships among the sentences as well. Word-based similarities among the sentences can be computed by simple similarity functions such as *cosine* in IR. Frequent information is the one that occurs in a lot of sentences, so it should be the source of the similarities between many sentences. In other words, if a sentence is similar to a lot of other

sentences in a document, it must contain frequent information that occurs at least in these other sentences. We argue that such a sentence is a good candidate to be included in a summary based on the old intuition that frequent information is the important or salient information in summarization. In our similarity graphs, these sentences must have strong connections to other sentences (nodes). They are easily detectable by looking at their central role (e.g. high degree) in the graph. We call this the *lexical centrality* value, or *LexRank*, of a sentence because it is a measure of the centrality of a sentence as a node in a graph, and the graph itself is constructed based on the lexical similarities among the sentences. After defining this similarity-based importance of a sentence in a graph, we improve our idea even more by looking at the similarities among the sentences that are two or more edges apart from each other. This leads to a recursive definition of sentence importance in which the importance of a sentence is dependent upon not only its similarities to other sentences but also the importances of the sentences that it is similar to. We implement this idea in terms of random walks in the similarity graph, where we propagate the similarity relation via the edges of the graph. Our motivation comes from the centrality methods used on the World Wide Web graph (Page et al., 1998) and the transitive nature of textual similarity we have assumed above.

With the success of graph-based centrality using random walks in text summarization, we extend our method to handle *focused* summarization in Chapter IV. Focused summarization is similar to generic summarization except that the summary needs to be based on a given query, a particular aspect of the general topic of the input document(s). We modify LexRank, and derive *biased LexRank* in which the centrality computation on a similarity graph also takes the query into account. Focused summarization is similar to a typical information retrieval scenario with

the exception that the desired output is a summary, or a ranked list of sentences, instead of a ranked list of documents. Our method is quite general so that it is applicable to similar query-based IR problems. Indeed, a graph-based method very similar in spirit to LexRank was later applied to document retrieval (Kurland and Lee, 2005). An interesting property of biased LexRank is that it simulates the idea of relevance feedback automatically and in a more principled way. The similarities of the sentences to the query are propagated via the similarities among the sentences. Therefore, the feedback information is taken into account *concurrently* with the initial retrieval scores in the overall process. There is no need for several rounds of relevance feedback and retrieval.

Propagation of similarity via random walks works well for the summarization problem. There are problems in IR where the concept of similarity plays a more direct role. Document clustering is one of them. The very definition of clustering is "putting *similar* things together". In Chapter V, we construct document similarity graphs where each node is a document. We again start with simple similarity measures and then "improve" them by taking random walks on the similarity graph into account as well. This gives us nonzero similarity values even for documents that are not immediate neighbors of each other. Using these random walk based similarities instead of the simple similarity functions traditionally used in document clustering, we get significant improvements in clustering accuracy.

Motivated by our document clustering experiments, we investigate document classification, the supervised counterpart of clustering, in Chapter VI. Nearest neighbor (NN) classifiers seem to be a good testbed to implement our ideas on graphs and random walks. NN classifiers are also based on similarity, so there is an underlying similarity graph structure in them. We show that by propagating the class informa-

tion in the graph via random walks, we can achieve better classification accuracy over the classical k-nearest neighbor (kNN) algorithm that rivals even the most state-of-the-art classification algorithms such as Support Vector Machines (SVM). Although the resultant algorithm, known as *harmonic functions*, was originally proposed in (Zhu et al., 2003), our treatment of it as the semi-supervised version of kNN and its application to text classification in this thesis is novel. It also relates well to our hypotheses and random walk ideas presented in the other chapters of this thesis.

Another contribution in this thesis starting with Chapter IV is using language modeling based similarity functions (see Chapter II) instead of the traditional *cosine* measure in the problems mentioned above. Language models can capture the asymmetry of the information content of two language strings as in Example I.1. Although the use of language model-based similarity measures to construct text similarity graphs was introduced by Kurland and Lee (2005), its application to summarization, document clustering and classification in this thesis is novel. The improvements we get only by using these measures instead of *cosine* without even applying our random walk ideas in document clustering (Chapter V) and document classification (Chapter VI) are striking. Using random walks yields additional improvement on top of them.

In Chapter II, we introduce the basic text similarity functions that we will repeatedly use in subsequent chapters. Most of what follows is published work. Chapter III was published as (Erkan and Radev, 2004b), which is an extended version of (Erkan and Radev, 2004a). Chapter IV is based on our experiments on the Document Understanding Conferences (DUC) data from 2005 and 2006 (Erkan, 2006b). An extended version is being submitted to the Natural Language Engineering (NLE) journal. The method presented in Chapter IV was also applied to the question answering problem

in (Otterbacher et al., 2005). Chapter V was published as (Erkan, 2006a). Chapter VI is being submitted to the ACM Conference on Information and Knowledge Management (CIKM).

CHAPTER II

BASIC TEXT SIMILARITY FUNCTIONS

2.1 Introduction

In this chapter, we introduce the basic definitions of the tools that will be reused repeatedly in constructing our methods in the subsequent chapters. The fundamental data structure we use is a graph where each node is a natural language string, and the edges (or links) between the nodes are induced by the specific similarity function defined on these strings. Formally, a graph is an ordered pair $G = (V, E)$ where V is the set of *nodes* and E is a set of pairs of distinct nodes. If an edge $(i, j) \in E$, we say i *links to* j, which is an indication of nodes i and j are somehow related depending on the semantic interpretation of the edges of the graph. A graph is *undirected* if $(i, j) \in E$ implies $(j, i) \in E$. Otherwise it is called *directed*. A graph is *weighted* if there is a function $w : E \rightarrow R^+$ associated with its edges. For weighted graphs, we say the graph is undirected only if the weighting function is symmetric.

In the next section, we will introduce several text similarity functions that will enable us to define an edge relation (with a weighting function) given a set of strings as the nodes of a graph. Some of these similarity functions are symmetric, leading to undirected graphs, while some are not necessarily symmetric.

2.2 Overlap and Cosine Similarity

Let us assume that we have two natural language strings s_1 and s_2. These can be as short as one word or as large as a novel or an entire library. However, we will only deal with sentences or documents of few hundred words in our problems in this thesis. Assuming we do not know much about the particular language that these strings are written in, the only thing we can do to assess the similarity between the two is to look at the words they have in common. The simplest similarity function, *overlap*, counts the number of words two strings have in common:

$$\text{overlap}(s_1, s_2) = \frac{\sum_{w \in s_1, s_2} 1}{min(|s_1|, |s_2|)} \tag{2.1}$$

where $|s|$ denotes the *length* of s or total number of words in it. The *overlap* function is normalized by the minimum of the lengths of two strings so that it always takes a value in $[0, 1]$. However, for strings of different length, *overlap* is not a good measure. Intuitively, the difference in the lengths of two strings should contribute to the similarity between them in a negative way. For an extreme example, consider a very long string that contains all the words in a language. The *overlap* between this string and any other string in the same language will take the maximum value 1.

A common technique in information retrieval is to represent strings as n-dimensional vectors where n is the total number of possible words in a particular language. We use $\text{tf}_{w,s}$ (term frequency) to denote the number of times word w occurs in string s. So a string s is represented as a vector $(\text{tf}_{w_1,s}, \text{tf}_{w_2,s}, \cdots, \text{tf}_{w_n,s})$ where each dimension holds the number of times the corresponding word occurs in s. The popular *cosine* function (Salton and McGill, 1983) is defined as follows:

$$\text{cosine}(s_1, s_2) = \frac{\sum_{w \in s_1, s_2} \text{tf}_{w,s_1} \cdot \text{tf}_{w,s_2}}{\|s_1\| \cdot \|s_2\|} \tag{2.2}$$

where $\|s\|$ denotes the L2 norm of the string vector s, that is,

$$\|s\| = \sqrt{\sum_{w \in s} (\mathrm{tf}_{w,s})^2} \tag{2.3}$$

As the name implies, the *cosine* similarity function measures the cosine of the angle between two string vectors. It takes a value in the $[0, 1]$ interval. It successfully accounts for the unseen words in a string. Unlike the *overlap* function, the words that occur only in one of the two strings hurt the similarity between them. It takes the maximum value 1 only if all the words in the language are represented with the same *ratio* (i.e. word count divided by the string length) in both strings.

Both similarity functions we have considered above give equal weight to every word in a language. However, our intuition about language tells us that some words are more important or carry more information than others. Common words (or *stop-words*) such as articles (e.g.*the, a*) or prepositions (e.g. *to, from, on*) occur almost in every sentence and should minimally contribute to the similarity between two strings. The frequency of the words seems to depend on the context as well. For example, it is normal to see the word *basketball* in a sports magazine, but the same word stands out in a financial article.

Suppose we have a weighting function $\phi(\cdot)$ defined on the words of a language possibly depending on a specific context. We can modify the string vectors and the *cosine* function as follows:

$$\text{weighted-cosine}(s_1, s_2) = \frac{\sum_{w \in s_1, s_2} \mathrm{tf}_{w,s_1} \cdot \mathrm{tf}_{w,s_2} \cdot \phi^2(w)}{\|s_1\| \cdot \|s_2\|} \tag{2.4}$$

where the L2 norm $\|s\|$ becomes

$$\|s\| = \sqrt{\sum_{w \in s} (\mathrm{tf}_{w,s})^2 \cdot \phi^2(w)} \tag{2.5}$$

One such weighting function we will consider in Chapter III is the *inverse document frequency* (IDF) which is based on the heuristic that the importance of a word is inversely proportional to the number of strings it occurs in a collection of strings.

2.3 Language Modeling Approach

A different approach to text similarity is to look at natural language strings as products of a probabilistic generative process. A *language model* is a probability distribution over the words of a language. It explains how likely it is for each word to occur in a string of that language. For example, we can talk about a general language model for English that gives us the probability of each word ever having been uttered in history. However, the language models in separate contexts or genres of text, such as news articles, everyday conversations or emails, are expected to be different.

2.3.1 Language Model Estimation

The language modeling approach treats a string as a sequence of words where each word is drawn independently from a multinomial probability distribution (language model). Language model *estimation* is the problem of computing a probability distribution from a given set of strings such that the probability of *generating* the same set of strings by independently drawing words from the computed distribution is as large as possible. Suppose we are given a string s, and we want to compute the language model that could have produced it. The *maximum likelihood estimation* (MLE) for s is the following probability distribution:

$$p_{\mathrm{ML}}(w|s) = \frac{\mathrm{tf}_{w,s}}{|s|} \tag{2.6}$$

that is, the probability of each word is equal to the ratio of the number of times it occurs in s to the total number of words in s. It can be shown that among all possible probability distributions, $p_{\mathrm{ML}}(w|s)$ is the one that assigns the maximum probability (likelihood) to s :

$$\begin{aligned} p_{\mathrm{ML}}(w|s) &= \operatorname*{argmax}_{\hat{p}} \frac{|s|!}{\prod_{w \in s}(\mathrm{tf}_{w,s})!} \prod_{w \in s} \hat{p}(w)^{\mathrm{tf}_{w,s}} \\ &= \operatorname*{argmax}_{\hat{p}} \prod_{w \in s} \hat{p}(w)^{\mathrm{tf}_{w,s}} \end{aligned} \tag{2.7}$$

The right hand side of Equation 2.7 is often called the *generation probability* of string s given the language model $\hat{p}$.

One problem with MLE is that it underestimates the probabilities of the words that occur rarely. In Equation 2.6, the probability of any word that does not occur in s is estimated to be zero. This is unrealistic for a probabilistic model of a language. Although certain words are very rare in certain contexts, assigning them zero probability is too harsh. Furthermore, the generation probability of a string becomes zero even if it has only one word with zero probability under a language model.

Assigning non-zero probabilities to unseen words in language model estimation is called *smoothing*. One of the oldest and the simplest smoothing methods is *additive* smoothing, also known as *Laplace* smoothing (Lidstone, 1920; Johnson, 1932; Jeffreys, 1961), where each word is assumed to have appeared σ times more than it did in reality:

$$p_{+\sigma}(w|s) = \frac{\mathrm{tf}_{w,s} + \sigma}{|s| + \sigma \cdot |\mathcal{V}|} \tag{2.8}$$

where $\mathcal{V}$ is the set of all possible words in the language. The special case where $\sigma = 1$ is called *add one* smoothing.

One problem with additive smoothing is that every unseen word is assigned the same probability. This may seem like the most reasonable thing to do if we have ab-

solutely no other information other than the string s. However, for a given language and a context, we usually know that certain unseen words are more probable than others. We can talk about a *back-off* language model from which we retrieve *default* probabilities of the unseen words. For example, if we are to compute a language model for a single news article about a specific event, a language model which was previously computed on a large set of news articles seems to be a good candidate for the probabilities of the unseen words.

Among other smoothing techniques proposed in the literature (Good, 1953; Church and Gale, 1991; Katz, 1987), here we focus on the general smoothing method proposed by (Jelinek and Mercer, 1980). Assume that we have a large corpus $\mathcal{C}$ of strings that we can compute a generic back-off language model from. Given a string s, *Jelinek-Mercer smoothing* is the following estimation:

$$p_{\mathrm{JM}}(w|s) = (1 - \lambda(s))p_{\mathrm{ML}}(w|s) + \lambda(s)p_{\mathrm{ML}}(w|\mathcal{C}) \tag{2.9}$$

where $\lambda(s)$ is the smoothing parameter. Equation 2.9 is an *interpolation* between the MLE of s and the more general MLE computed from a large corpus $\mathcal{C}$.

Note that $\lambda(s)$ is a function that can take different values on different strings. The simplest technique is to choose a constant function. However, if we have reason to believe that the MLE we compute from a string s is reliable enough, we may want to choose a smaller $\lambda(s)$ for this particular string so that the effect of the corpus MLE in the interpolation is minimal. One method proposed in (MacKay and Peto, 1995) is motivated within a Bayesian framework. A language model is assumed to be a multinomial distribution whose conjugate prior is the Dirichlet distribution. In this analysis, the smoothing parameter in Equation 2.9 is chosen as

$$\lambda(s) = \frac{\mu}{|s| + \mu} \tag{2.10}$$

where μ is the smoothing parameter which has an effect similar to σ in additive smoothing. Note that as the input string s gets larger $\lambda(s)$ gets smaller, that is, for longer strings the estimation relies more on the input string than the back-off language model. This is intuitive since longer strings give more evidence for their underlying language model, so we should be more confident of the MLE we compute from them compared to the MLEs of shorter strings. With this parameter setting, the final smoothed language model becomes

$$p_{\mathrm{Dir}}(w|s) = \frac{\mathrm{tf}_{w,s} + \mu \cdot p_{\mathrm{ML}}(w|\mathcal{C})}{|s| + \mu} \tag{2.11}$$

2.3.2 Language Model Based Similarity

In this section, we explain how we can compute the similarity between two strings once we have estimated their underlying language models. Equation 2.7 already suggests one. Assuming we have a language model computed from a string s_1, we can compute the likelihood (generation probability) of another string s_2 under this language model:

$$p_{\mathrm{gen}}(s_2|s_1) = \prod_{w \in s_2} p(w|s_1)^{\mathrm{tf}_{w,s_2}} \tag{2.12}$$

where $p(w|s_1)$ is a language model computed from s_1 by using one of the smoothed estimation methods we have introduced in the previous section. This quantity is used in solving many NLP problems. In speech recognition, the area that actually introduced language models, the generation probabilities of the candidate output strings under a pre-computed generic language model are used to assess which candidate is more likely to occur. In machine translation, the candidate translations in the target language are treated similarly. However, the generation probabilities used in information retrieval are more relevant to the similarity concept we are dealing with

here. For example, in document retrieval, given a query string q, the relevance of each document d_i in the collection is assessed by the generation probability of q from d_i, $p_{\text{gen}}(q|d_i)$. In other words, documents are ranked by the likelihoods that their underlying language models could have produced the query. This *language modeling approach to information retrieval* has proven to be very successful in recent years compared to more classical retrieval methods such as computing the cosine similarity between the query and each document (Ponte and Croft, 1998; Lafferty and Zhai, 2001).

Since all the word probabilities are multiplied with each other, the generation probabilities for longer documents tend to be much smaller. This does not create any problems in a typical document retrieval scenario because given a collection of document language models, only the generation probabilities for a single query are compared against each other. However, if we were asked which of the two queries is more likely for a given document, then the shorter query string would have much more chance. Therefore it is troublesome to propose Equation 2.12 as a general similarity function. Its bias towards shorter documents makes the similarity values across different pairs of strings incomparable. One ad-hoc solution proposed by (Lavrenko et al., 2002) is to take the geometric mean of the word probabilities to normalize the generation probability by the string length:

$$p_{\text{norm}}(s_2|s_1) = \left(\prod_{w \in s_2} p(w|s_1)^{\text{tf}_{w,s_2}} \right)^{\frac{1}{|s_2|}} \tag{2.13}$$

An alternative approach to computing the generation probability $p_{\text{gen}}(s_2|s_1)$ is to compute the language model of s_2 as well, and then directly compare it against the language model of s_1. This gives us a measure of the similarity between two probability distributions. One popular method to compare two probability distributions is

KL divergence (Cover and Thomas, 1991), which measures how *different* two probability distributions are. Given two probability distributions p and r, KL divergence, also known as *relative entropy*, is defined as

$$\mathrm{KL}(p\|r) = \sum_w p(w) \log \frac{p(w)}{r(w)} \tag{2.14}$$

If p and r are identical distributions, then $\mathrm{KL}(p\|r)$ equals zero. Otherwise, it can take values up to ∞ depending on how different two distributions are. Since KL divergence measures the *distance*, not the similarity, between two distributions, the negative KL divergence is usually used as a similarity measure.

The use of KL divergence on language models as a similarity function for NLP applications was introduced in (Lee, 1997). Later it has been popularized as a retrieval method in the language modeling approach to information retrieval. Suppose we are again given a query q, and we want to rank the documents d_i in a collection based on this query. The KL divergence based document retrieval method uses the negative KL divergence between the query language model and the document language model, $-\mathrm{KL}(p(\cdot|q)\|p(\cdot|d_i))$, to score each document d_i (Lafferty and Zhai, 2001). The simplest version uses MLE for the query language model:

$$-\mathrm{KL}(p_{\mathrm{ML}}(\cdot|q)\|p(\cdot|d_i)) = \sum_w p_{\mathrm{ML}}(w|q) \log \frac{p(w|d_i)}{p_{\mathrm{ML}}(w|q)} \tag{2.15}$$

The document language model $p(w|d_i)$ can be computed by any smoothing method. It can be shown that Equation 2.15 yields exactly the same *ranking* of the documents as the generation probability in Equation 2.12 does. So, from the document retrieval perspective, there is no practical difference between using KL divergence and generation probabilities. However, KL divergence has some useful properties when used as a similarity measure. Let us again have two strings s_1 and s_2, and we want to compute the similarity between them using Equation 2.15. A popular

similarity function based on KL divergence is $exp(-\mathrm{KL}(\cdot\|\cdot))$. One advantage of this function is that it takes values in the interval $[0, 1]$, which is a desirable property for a similarity function. Another interesting property, as pointed out by (Kurland and Lee, 2005), becomes more obvious with the following derivation:

$$\begin{aligned}
\mathrm{sim}_{\mathrm{KL}}(s_1, s_2) &= exp(-\mathrm{KL}(p_{\mathrm{ML}}(\cdot|s_2)\|p(\cdot|s_1)) \\
&= \prod_w \left(\frac{p(w|s_1)}{p_{\mathrm{ML}}(w|s_2)}\right)^{p_{\mathrm{ML}}(w|s_2)} \\
&= \prod_{w\in s_2} p(w|s_1)^{p_{\mathrm{ML}}(w|s_2)} \cdot \prod_{w\in s_2} p_{\mathrm{ML}}(w|s_2)^{-p_{\mathrm{ML}}(w|s_2)} \\
&= \prod_{w\in s_2} \left(p(w|s_1)^{\mathrm{tf}_{w,s_2}}\right)^{\frac{1}{|s_2|}} \cdot \prod_{w\in s_2} p_{\mathrm{ML}}(w|s_2)^{-p_{\mathrm{ML}}(w|s_2)} \qquad (2.16)
\end{aligned}$$

Note that the first factor in Equation 2.16 is exactly the same as Equation 2.13. The second factor depends only on the language model of s_2. In fact, it can be shown that the second factor is equal to $exp(H(p_{\mathrm{ML}}(\cdot|s_2))$ where H is the *entropy* function in information theory. Therefore, KL divergence based similarity is already normalized by the string length, solving the problem of length bias of Equation 2.12 in a principled way.

None of the language model based similarity functions we have introduced in this section is a symmetric function. There have been attempts to introduce symmetric distance/similarity functions based on probability distributions. For example, a symmetric distance measure based on KL divergence is the Jensen-Shannon divergence (Lin, 1991) defined as

$$\mathrm{JS}(p\|r) = \frac{1}{2}\left(\mathrm{KL}(p\|\frac{p+r}{2}) + \mathrm{KL}(r\|\frac{p+r}{2})\right) \qquad (2.17)$$

However, we will argue that having asymmetric similarity functions is actually more intuitive for text similarity. Although the amount of *shared* information between two strings is always the same for both of them, the *unshared* information they contain

might be quite different. A motivating example is the sentence pair in Example I.1 which we repeat here:

Example II.1.

S_1: *John loves Mary.*

S_2: *John loves Mary, but she does not know this.*

S_2 contains more information than S_1 does. The language modeling framework can capture this phenomenon. From the probabilistic and generative perspective, S_1 should have a higher chance of being generated from S_2 than vice versa since S_2 contains all the information S_1 has. This is correctly predicted by the generation probabilities. Note that $p_{\text{ML}}(S_2|S_1) = 0$ for the unsmoothed language model $p_{\text{ML}}(\cdot|S_1)$ while $p_{\text{ML}}(S_1|S_2)$ is non-zero. $p_{\text{gen}}(S_1|S_2) > p_{\text{gen}}(S_2|S_1)$ holds under all smoothing technique variations we have considered above. Same holds for $sim_{\text{KL}}(S_1, S_2) > sim_{\text{KL}}(S_2, S_1)$.

As a final note on similarity, we want to point out that all the similarity functions we have considered in this chapter assume that a language string is a mere set of words without paying attention to the order of the words in the strings. In other words, these similarity functions are insensitive to permutations of words in a string. Although this may seem to be a harsh oversimplification, almost all of the popular approaches to the problems we consider in this thesis use similar similarity functions. This is mostly because the current state of the more syntactically aware similarity functions is not good enough for practical purposes. Using the simple functions above is both more efficient and more effective most of the time. Moreover, our purpose in this thesis is to use such simple, efficient and language-independent similarity functions to build a deeper understanding of text similarity on top of them.

CHAPTER III

GRAPH-BASED CENTRALITY FOR SENTENCE RETRIEVAL AND EXTRACTIVE SUMMARIZATION

3.1 Introduction

In recent years, natural language processing (NLP) has moved to a very firm mathematical foundation. Many problems in NLP, e.g., parsing (Collins, 1997), word sense disambiguation (Yarowsky, 1995), and automatic paraphrasing (Barzilay and Lee, 2003) have benefited significantly by the introduction of robust statistical techniques. Recently, robust graph-based methods for NLP have also been gaining a lot of interest, e.g., in word clustering (Brew and im Walde, 2002) and prepositional phrase attachment (Toutanova et al., 2004). In this chapter, we will take graph-based methods in NLP one step further. We will discuss how random walks on sentence-based graphs can help in text summarization.

Text summarization is the process of automatically creating a compressed version of a given text that provides useful information for the user. The information content of a summary depends on users' needs. Topic-oriented summaries focus on a user's topic of interest, and extract the information in the text that is related to the specified topic. On the other hand, generic summaries try to cover as much of the information content as possible, preserving the general topical organization of the original text.

In this chapter, we focus on multi-document extractive generic text summarization, where the goal is to produce a summary of multiple documents about the same, but unspecified topic.

Extractive summarization produces summaries by choosing a subset of the sentences in the original document(s). This contrasts with abstractive summarization, where the information in the text is rephrased. Although summaries produced by humans are typically not extractive, most of the summarization research today is on extractive summarization. Purely extractive summaries often give better results compared to automatic abstractive summaries. This is due to the fact that the problems in abstractive summarization, such as semantic representation, inference and natural language generation, are relatively harder compared to a data-driven approach such as sentence extraction. In fact, truly abstractive summarization has not reached to a mature stage today. Existing abstractive summarizers often depend on an extractive preprocessing component. The output of the extractor is cut and pasted, or compressed to produce the abstract of the text (Witbrock and Mittal, 1999; Jing, 2002; Knight and Marcu, 2000). SUMMONS (Radev and McKeown, 1998) is an example of a multi-document summarizer which extracts and combines information from multiple sources and passes this information to a language generation component to produce the final summary.

Early research on extractive summarization is based on simple heuristic features of the sentences such as their position in the text, the overall frequency of the words they contain, or some key phrases indicating the importance of the sentences (Baxendale, 1958; Edmundson, 1969; Luhn, 1958). A commonly used measure to assess the importance of the words in a sentence is the *inverse document frequency*, or idf,

which is defined by the formula (Sparck-Jones,):

$$\text{idf}_i = \log\Big(\frac{N}{n_i}\Big) \tag{3.1}$$

where N is the total number of the documents in a collection, and n_i is the number of documents in which word i occurs. For example, the words that are likely to occur in almost every document (e.g. articles "a" and "the") have idf values close to zero while rare words (e.g. medical terms, proper nouns) typically have higher idf values.

More advanced techniques also consider the relation between sentences or the discourse structure by using synonyms of the words or anaphora resolution (Mani and Bloedorn, 1997; Barzilay and Elhadad, 1999). Researchers have also tried to integrate machine learning into summarization as more features have been proposed and more training data have become available (Kupiec et al., 1995; Lin, 1999; Osborne, 2002; Daumé III and Marcu, 2004).

Our summarization approach in this chapter is to assess the *centrality* of each sentence in a cluster and extract the most important ones to include in the summary. We investigate different ways of defining the lexical centrality principle in multi-document summarization, which measures centrality in terms of lexical properties of the sentences.

In Section 3.2, we present centroid-based summarization, a well-known method for judging sentence centrality. Then we introduce three new measures for centrality, Degree, LexRank with threshold, and continuous LexRank, inspired from the "prestige" concept in social networks. We propose a graph representation of a document cluster, where vertices represent the sentences and edges are defined in terms of the similarity relation between pairs of sentences. This representation enables us to make use of several centrality heuristics defined on graphs. We compare our new methods with centroid-based summarization using a feature-based generic summarization

toolkit, MEAD, and show that our new features outperform Centroid in most of the cases. Test data for our experiments are taken from 2003 and 2004 summarization evaluations of Document Understanding Conferences (DUC) to compare our system with other state-of-the-art summarization systems and human performance as well.

3.2 Sentence Centrality and Centroid-based Summarization

Extractive summarization works by choosing a subset of the sentences in the original documents. This process can be viewed as identifying the most *central* sentences in a (multi-document) cluster that give the necessary and sufficient amount of information related to the main theme of the cluster. Centrality of a sentence is often defined in terms of the centrality of the words that it contains. A common way of assessing word centrality is to look at the centroid of the document cluster in a vector space. The centroid of a cluster is a pseudo-document which consists of words that have tf×idf scores above a predefined threshold, where tf is the frequency of a word in the cluster, and idf values are typically computed over a much larger and similar genre data set. In centroid-based summarization (Radev et al., 2000), the sentences that contain more words from the centroid of the cluster are considered as central (Algorithm 1). This is a measure of how close the sentence is to the centroid of the cluster. Centroid-based summarization has given promising results in the past, and it has resulted in the first web-based multi-document summarization system[1] (Radev et al., 2001).

[1] `http://www.newsinessence.com`

input : An array S of n sentences, cosine threshold t
output: An array C of Centroid scores

```
1  Hash WordHash;
2  Array C;
3  /* compute tf×idf scores for each word */
4  for i ← 1 to n do
5      foreach word w of S[i] do
6          WordHash{w}{"tfidf"} = WordHash{w}{"tfidf"} + idf{w};
7      end
8  end
9  /* construct the centroid of the cluster */
10 /* by taking the words that are above the threshold*/
11 foreach word w of WordHash do
12     if WordHash{w}{"tfidf"} > t then
13         WordHash{w}{"centroid"} = WordHash{w}{"tfidf"};
14     end
15     else
16         WordHash{w}{"centroid"} = 0;
17     end
18 end
19 /* compute the score for each sentence */
20 for i ← 1 to n do
21     C[i] = 0;
22     foreach word w of S[i] do
23         C[i] = C[i] + WordHash{w}{"centroid"};
24     end
25 end
26 return C;
```

Algorithm 1: Computing Centroid scores.

3.3 Centrality-based Sentence Salience

In this section, we propose several other criteria to assess sentence salience. All of our approaches are based on the concept of *prestige*[2] in social networks, which has also inspired many ideas in computer networks and information retrieval. A social network is a mapping of relationships between interacting entities (e.g. people, organizations, computers). Social networks are represented as graphs, where the nodes represent the entities and the links represent the relations between the nodes.

A cluster of documents can be viewed as a network of sentences that are related to each other. Some sentences are more similar to each other while some others may share only a little information with the rest of the sentences. We hypothesize that the sentences that are similar to many of the other sentences in a cluster are more central (or *salient*) to the topic. There are two points to clarify in this definition of centrality. First is how to define similarity between two sentences. Second is how to compute the overall centrality of a sentence given its similarity to other sentences.

To define similarity, we use the bag-of-words model to represent each sentence as an N-dimensional vector, where N is the number of all possible words in the target language. For each word that occurs in a sentence, the value of the corresponding dimension in the vector representation of the sentence is the number of occurrences of the word in the sentence times the idf of the word. The similarity between two sentences is then defined by the cosine between two corresponding vectors:

$$\text{idf-modified-cosine}(x, y) = \frac{\sum_{w \in x,y} \text{tf}_{w,x}\text{tf}_{w,y}(\text{idf}_w)^2}{\sqrt{\sum_{x_i \in x}(\text{tf}_{x_i,x}\text{idf}_{x_i})^2} \times \sqrt{\sum_{y_i \in y}(\text{tf}_{y_i,y}\text{idf}_{y_i})^2}} \tag{3.2}$$

where $\text{tf}_{w,s}$ is the number of occurrences of the word w in the sentence s.

A cluster of documents may be represented by a cosine similarity matrix where

[2] "Prestige" and "centrality" stand for the same concept with the difference that the former is often defined for directed graphs whereas the latter is defined for undirected graphs.

each entry in the matrix is the similarity between the corresponding sentence pair. Figure 3.1 shows a subset of a cluster used in DUC 2004, and the corresponding cosine similarity matrix. Sentence ID dXsY indicates the Y^{th} sentence in the X^{th} document. This matrix can also be represented as a weighted graph where each edge shows the cosine similarity between a pair of sentence (Figure 3.2). In the following sections, we discuss several ways of computing sentence centrality using the cosine similarity matrix and the corresponding graph representation.

3.3.1 Degree Centrality

In a cluster of related documents, many of the sentences are expected to be somewhat similar to each other since they are all about the same topic. This can be seen in Figure 3.1 where the majority of the values in the similarity matrix are nonzero. Since we are interested in *significant* similarities, we can eliminate some low values in this matrix by defining a threshold so that the cluster can be viewed as an (undirected) graph, where each sentence of the cluster is a node, and significantly similar sentences are connected to each other. Figure 3.3 shows the graphs that correspond to the adjacency matrices derived by assuming the pair of sentences that have a similarity above $0.1, 0.2$, and 0.3, respectively, in Figure 3.1 are similar to each other. Note that there should also be self links for all of the nodes in the graphs since every sentence is trivially similar to itself. Although we omit the self links for readability, the arguments in the following sections assume that they exist.

A simple way of assessing sentence centrality by looking at the graphs in Figure 3.3 is to count the number of similar sentences for each sentence. We define *degree centrality* of a sentence as the degree of the corresponding node in the similarity graph. As seen in Table 3.1, the choice of cosine threshold dramatically influences the

interpretation of centrality. Too low thresholds may mistakenly take weak similarities into consideration while too high thresholds may lose many of the similarity relations in a cluster.

ID	Degree (0.1)	Degree (0.2)	Degree (0.3)
d1s1	5	4	2
d2s1	7	4	2
d2s2	2	1	1
d2s3	6	3	1
d3s1	5	2	1
d3s2	7	5	1
d3s3	2	2	1
d4s1	9	6	1
d5s1	5	4	2
d5s2	6	4	1
d5s3	5	2	2

Table 3.1: Degree centrality scores for the graphs in Figure 3.3. Sentence d4s1 is the most central sentence for thresholds 0.1 and 0.2.

3.3.2 Eigenvector Centrality and LexRank

When computing degree centrality, we have treated each edge as a *vote* to determine the overall centrality value of each node. This is a totally democratic method where each vote counts the same. However, in many types of social networks, not all of the relationships are considered equally important. As an example, consider a social network of people that are connected to each other with the friendship relation. The prestige of a person does not only depend on how many friends he has, but also depends on *who* his friends are.

The same idea can be applied to extractive summarization as well. Degree centrality may have a negative effect in the quality of the summaries in some cases where several unwanted sentences vote for each other and raise their centrality. As an extreme example, consider a noisy cluster where all the documents are related to each other, but only one of them is about a somewhat different topic. Obviously, we would not want any of the sentences in the unrelated document to be included

SNo	ID	Text
1	d1s1	Iraqi Vice President Taha Yassin Ramadan announced today, Sunday, that Iraq refuses to back down from its decision to stop cooperating with disarmament inspectors before its demands are met.
2	d2s1	Iraqi Vice president Taha Yassin Ramadan announced today, Thursday, that Iraq rejects cooperating with the United Nations except on the issue of lifting the blockade imposed upon it since the year 1990.
3	d2s2	Ramadan told reporters in Baghdad that "Iraq cannot deal positively with whoever represents the Security Council unless there was a clear stance on the issue of lifting the blockade off of it.
4	d2s3	Baghdad had decided late last October to completely cease cooperating with the inspectors of the United Nations Special Commission (UNSCOM), in charge of disarming Iraq's weapons, and whose work became very limited since the fifth of August, and announced it will not resume its cooperation with the Commission even if it were subjected to a military operation.
5	d3s1	The Russian Foreign Minister, Igor Ivanov, warned today, Wednesday against using force against Iraq, which will destroy, according to him, seven years of difficult diplomatic work and will complicate the regional situation in the area.
6	d3s2	Ivanov contended that carrying out air strikes against Iraq, who refuses to cooperate with the United Nations inspectors, "will end the tremendous work achieved by the international group during the past seven years and will complicate the situation in the region."
7	d3s3	Nevertheless, Ivanov stressed that Baghdad must resume working with the Special Commission in charge of disarming the Iraqi weapons of mass destruction (UNSCOM).
8	d4s1	The Special Representative of the United Nations Secretary-General in Baghdad, Prakash Shah, announced today, Wednesday, after meeting with the Iraqi Deputy Prime Minister Tariq Aziz, that Iraq refuses to back down from its decision to cut off cooperation with the disarmament inspectors.
9	d5s1	British Prime Minister Tony Blair said today, Sunday, that the crisis between the international community and Iraq "did not end" and that Britain is still "ready, prepared, and able to strike Iraq."
10	d5s2	In a gathering with the press held at the Prime Minister's office, Blair contended that the crisis with Iraq "will not end until Iraq has absolutely and unconditionally respected its commitments" towards the United Nations.
11	d5s3	A spokesman for Tony Blair had indicated that the British Prime Minister gave permission to British Air Force Tornado planes stationed in Kuwait to join the aerial bombardment against Iraq.

	1	2	3	4	5	6	7	8	9	10	11
1	1.00	0.45	0.02	0.17	0.03	0.22	0.03	0.28	0.06	0.06	0.00
2	0.45	1.00	0.16	0.27	0.03	0.19	0.03	0.21	0.03	0.15	0.00
3	0.02	0.16	1.00	0.03	0.00	0.01	0.03	0.04	0.00	0.01	0.00
4	0.17	0.27	0.03	1.00	0.01	0.16	0.28	0.17	0.00	0.09	0.01
5	0.03	0.03	0.00	0.01	1.00	0.29	0.05	0.15	0.20	0.04	0.18
6	0.22	0.19	0.01	0.16	0.29	1.00	0.05	0.29	0.04	0.20	0.03
7	0.03	0.03	0.03	0.28	0.05	0.05	1.00	0.06	0.00	0.00	0.01
8	0.28	0.21	0.04	0.17	0.15	0.29	0.06	1.00	0.25	0.20	0.17
9	0.06	0.03	0.00	0.00	0.20	0.04	0.00	0.25	1.00	0.26	0.38
10	0.06	0.15	0.01	0.09	0.04	0.20	0.00	0.20	0.26	1.00	0.12
11	0.00	0.00	0.00	0.01	0.18	0.03	0.01	0.17	0.38	0.12	1.00

Figure 3.1: Intra-sentence cosine similarities in a subset of cluster d1003t from DUC 2004. Source: Agence France Presse (AFP) Arabic Newswire (1998). Manually translated to English.

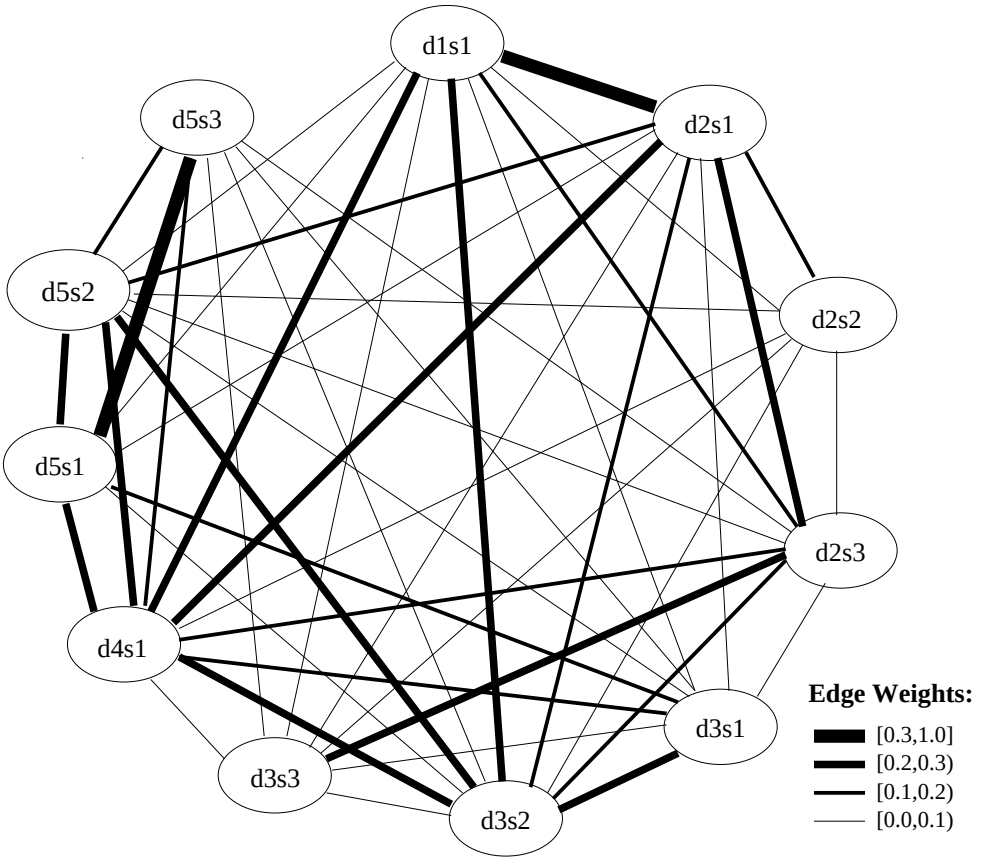


Figure 3.2: Weighted cosine similarity graph for the cluster in Figure 3.1.

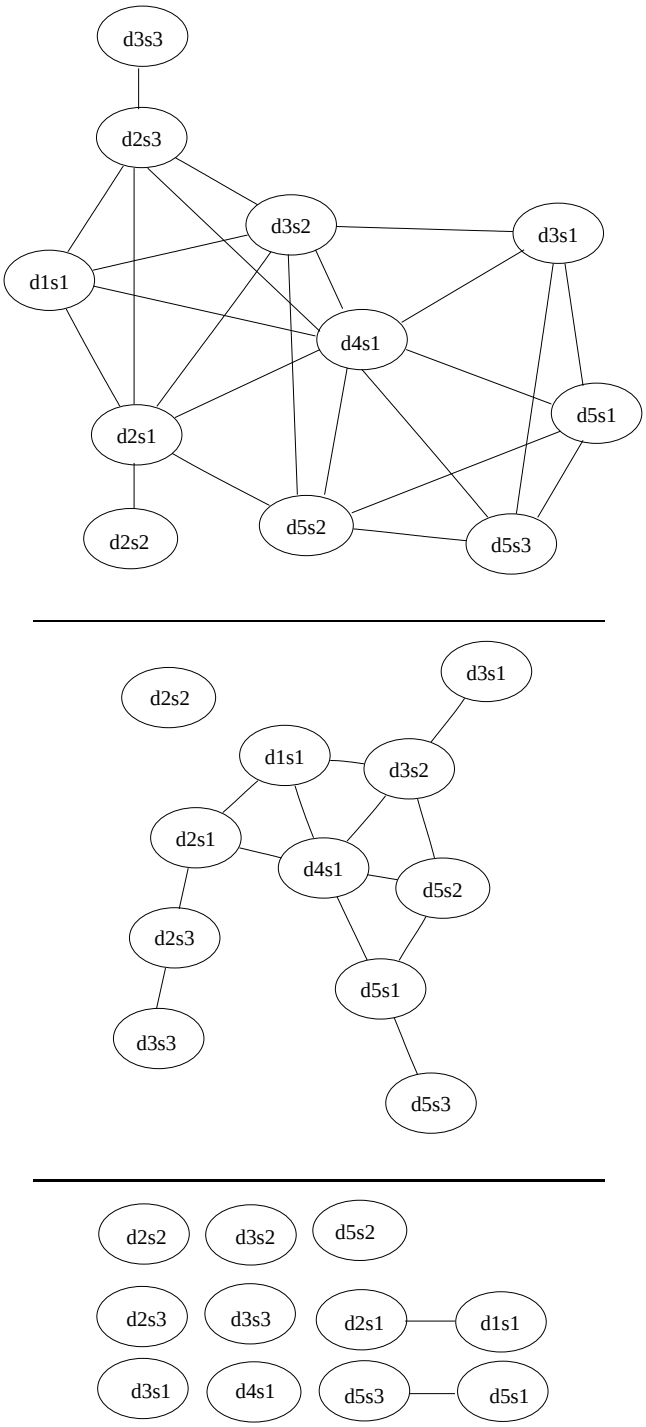


Figure 3.3: Similarity graphs that correspond to thresholds 0.1, 0.2, and 0.3, respectively, for the cluster in Figure 3.1.

in a generic summary of the cluster. However, suppose that the unrelated document contains some sentences that are very prestigious considering only the votes in that document. These sentences will get artificially high centrality scores by the local votes from a specific set of sentences. This situation can be avoided by considering where the votes come from and taking the centrality of the *voting* nodes into account in weighting each vote. A straightforward way of formulating this idea is to consider every node having a centrality value and distributing this centrality to its neighbors. This formulation can be expressed by the equation

$$p(u) = \sum_{v \in adj[u]} \frac{p(v)}{deg(v)} \tag{3.3}$$

where $p(u)$ is the centrality of node u, $adj[u]$ is the set of nodes that are adjacent to u, and $deg(v)$ is the degree of the node v. Equivalently, we can write Equation 3.3 in the matrix notation as

$$\mathbf{p} = \mathbf{B}^{\mathrm{T}}\mathbf{p} \tag{3.4}$$

or

$$\mathbf{p}^{\mathrm{T}}\mathbf{B} = \mathbf{p}^{\mathrm{T}} \tag{3.5}$$

where the matrix $\mathbf{B}$ is obtained from the adjacency matrix of the similarity graph by dividing each element by the corresponding row sum:

$$\mathbf{B}(i,j) = \frac{\mathbf{A}(i,j)}{\sum_k \mathbf{A}(i,k)} \tag{3.6}$$

Note that a row sum is equal to the degree of the corresponding node. Since every sentence is similar at least to itself, all row sums are nonzero. Equation 3.5 states that $\mathbf{p}^{\mathrm{T}}$ is the left eigenvector of the matrix $\mathbf{B}$ with the corresponding eigenvalue of 1. To guarantee that such an eigenvector exists and can be uniquely identified and computed, we need some mathematical foundations.

A *stochastic* matrix, $\mathbf{X}$, is the transition matrix of a Markov chain. An element $\mathbf{X}(i,j)$ of a stochastic matrix specifies the transition probability from state i to state j in the corresponding Markov chain. By the probability axioms, all rows of a stochastic matrix should add up to 1. $\mathbf{X^n}(i,j)$ gives the probability of reaching from state i to state j in n transitions. A Markov chain with the stochastic matrix $\mathbf{X}$ converges to a stationary distribution if

$$\lim_{n\to\infty} \mathbf{X^n} = \mathbf{1}^{\mathrm{T}}\mathbf{r} \tag{3.7}$$

where $\mathbf{1} = (1, 1, ..., 1)$, and the vector $\mathbf{r}$ is called the stationary distribution of the Markov chain. An intuitive interpretation of the stationary distribution can be understood by the concept of a random walk. Each element of the vector $\mathbf{r}$ gives the asymptotic probability of ending up in the corresponding state in the long run regardless of the starting state. A Markov chain is *irreducible* if any state is reachable from any other state, i.e. for all i, j there exists an n such that $\mathbf{X^n}(i,j) \neq 0$. A Markov chain is *aperiodic* if for all i, $gcd\{n : X^n(i,i) > 0\} = 1$. By the Perron-Frobenius theorem (Seneta, 1981), an irreducible and aperiodic Markov chain is guaranteed to converge to a unique stationary distribution. If a Markov chain has reducible or periodic components, a random walker may get stuck in these components and never visit the other parts of the graph.

Since the similarity matrix $\mathbf{B}$ in Equation 3.4 satisfies the properties of a stochastic matrix, we can treat it as a Markov chain. The centrality vector $\mathbf{p}$ corresponds to the stationary distribution of $\mathbf{B}$. However, we need to make sure that the similarity matrix is always irreducible and aperiodic. To solve this problem, (Page et al., 1998) suggest reserving some low probability for jumping to any node in the graph. This way the random walker can "escape" from periodic or disconnected components, which makes the graph irreducible and aperiodic. If we assign a uniform probability

for jumping to any node in the graph, we are left with the following modified version of Equation 3.3, which is known as PageRank,

$$p(u) = \frac{d}{N} + (1-d) \sum_{v \in adj[u]} \frac{p(v)}{deg(v)} \tag{3.8}$$

where N is the total number of nodes in the graph, and d is a "damping factor", which is typically chosen in the interval $[0.1, 0.2]$ (Brin and Page, 1998). Equation 3.8 can be written in the matrix form as

$$\mathbf{p} = [d\mathbf{U} + (1-d)\mathbf{B}]^{\mathrm{T}}\mathbf{p} \tag{3.9}$$

where $\mathbf{U}$ is a square matrix with all elements being equal to $1/N$. The transition kernel $[d\mathbf{U} + (1-d)\mathbf{B}]$ of the resulting Markov chain is a mixture of two kernels $\mathbf{U}$ and $\mathbf{B}$. A random walker on this Markov chain chooses one of the adjacent states of the current state with probability $1-d$, or jumps to any state in the graph, including the current state, with probability d. The PageRank formula was first proposed for computing web page prestige, and still serves as the underlying mechanism behind the Google search engine.

The convergence property of Markov chains also provides us with a simple iterative algorithm, called power method, to compute the stationary distribution (Algorithm 2). The algorithm starts with a uniform distribution. At each iteration, the eigenvector is updated by multiplying with the transpose of the stochastic matrix. Since the Markov chain is irreducible and aperiodic, the algorithm is guaranteed to terminate.

Unlike the original PageRank method, the similarity graph for sentences is undirected since cosine similarity is a symmetric relation. However, this does not make any difference in the computation of the stationary distribution. We call this new measure of sentence similarity *lexical PageRank*, or *LexRank*. Algorithm 3 summa-

input : A stochastic, irreducible and aperiodic matrix $\mathbf{M}$
input : matrix size N, error tolerance ϵ
output: eigenvector $\mathbf{p}$

$\mathbf{p}_0 = \frac{1}{N}\mathbf{1}$;
t=0;
repeat
    t=t+1;
    $\mathbf{p}_t = \mathbf{M}^{\mathrm{T}}\mathbf{p}_{t-1}$;
    $\delta = ||\mathbf{p}_t - \mathbf{p}_{t-1}||$;
until $\delta < \epsilon$;
return $\mathbf{p}_t$;

Algorithm 2: Power Method for computing the stationary distribution of a Markov chain.

rizes how to compute LexRank scores for a given set of sentences. Note that Degree centrality scores are also computed (in the *Degree* array) as a side product of the algorithm. Table 3.2 shows the LexRank scores for the graphs in Figure 3.3 setting the damping factor to 0.85. For comparison, Centroid score for each sentence is also shown in the table. All the numbers are normalized so that the highest ranked sentence gets the score 1. It is obvious from the figures that threshold choice affects the LexRank rankings of some sentences.

3.3.3 Continuous LexRank

The similarity graphs we have constructed to compute Degree centrality and LexRank are unweighted. This is due to the binary discretization we perform on the cosine matrix using an appropriate threshold. As in all discretization operations, this means an information loss. One improvement over LexRank can be obtained by making use of the *strength* of the similarity links. If we use the cosine values directly to construct the similarity graph, we usually have a much denser but weighted graph (Figure 3.2). We can normalize the row sums of the corresponding transition matrix so that we have a stochastic matrix. The resultant equation is a modified version of LexRank

```
MInputAn array S of n sentences, cosine threshold t  output: An array L of LexRank scores
Array CosineMatrix[n][n];
Array Degree[n];
Array L[n];
for i ← 1 to n do
    for j ← 1 to n do
        CosineMatrix[i][j] = idf-modified-cosine(S[i],S[j]);
        if CosineMatrix[i][j] > t then
            CosineMatrix[i][j] = 1;
            Degree[i] + +;
        end
        else
            CosineMatrix[i][j] = 0;
        end
    end
end
for i ← 1 to n do
    for j ← 1 to n do
        CosineMatrix[i][j] = CosineMatrix[i][j]/Degree[i];
    end
end
L = PowerMethod(CosineMatrix,n,ϵ);
return L;
```

Algorithm 3: Computing LexRank scores.

ID	LR (0.1)	LR (0.2)	LR (0.3)	Centroid
d1s1	0.6007	0.6944	1.0000	0.7209
d2s1	0.8466	0.7317	1.0000	0.7249
d2s2	0.3491	0.6773	1.0000	0.1356
d2s3	0.7520	0.6550	1.0000	0.5694
d3s1	0.5907	0.4344	1.0000	0.6331
d3s2	0.7993	0.8718	1.0000	0.7972
d3s3	0.3548	0.4993	1.0000	0.3328
d4s1	1.0000	1.0000	1.0000	0.9414
d5s1	0.5921	0.7399	1.0000	0.9580
d5s2	0.6910	0.6967	1.0000	1.0000
d5s3	0.5921	0.4501	1.0000	0.7902

Table 3.2: LexRank scores for the graphs in Figure 3.3. All the values are normalized so that the largest value of each column is 1. Sentence d4s1 is the most central page for thresholds 0.1 and 0.2.

for weighted graphs:

$$p(u) = \frac{d}{N} + (1-d) \sum_{v \in adj[u]} \frac{\text{idf-modified-cosine}(u,v)}{\sum_{z \in adj[v]} \text{idf-modified-cosine}(z,v)} p(v) \tag{3.10}$$

This way, while computing LexRank for a sentence, we multiply the LexRank values of the linking sentences by the weights of the links. Weights are normalized by the row sums, and the damping factor d is added for the convergence of the method.

3.3.4 Centrality vs. Centroid

Graph-based centrality has several advantages over Centroid. First of all, it accounts for information subsumption among sentences. If the information content of a sentence subsumes another sentence in a cluster, it is naturally preferred to include the one that contains more information in the summary. The degree of a node in the cosine similarity graph is an indication of how much common information the sentence has with other sentences. Sentence d4s1 in Figure 3.1 gets the highest score since it almost subsumes the information in the first two sentences of the cluster and has some common information with others. Another advantage of our proposed approach is that it prevents unnaturally high idf scores from boosting up the score of a sentence that is unrelated to the topic. Although the frequency of the words are taken into account while computing the Centroid score, a sentence that contains many rare words with high idf values may get a high Centroid score even if the words do not occur elsewhere in the cluster.

3.4 Experimental Setup

In this section, we describe the data set, the evaluation metric and the summarization system we used in our experiments.

3.4.1 Data Set and Evaluation Method

We used DUC 2003 and 2004 data sets in our experiments. Task 2 of both DUC 2003 and 2004 involve generic summarization of news documents clusters. There are a total of 30 clusters in DUC 2003 and 50 clusters in DUC 2004. In addition to these two tasks, we used two more data sets from Task 4 of DUC 2004, which involves cross-lingual generic summarization. First set (Task 4a) is composed of Arabic-to-English machine translations of 24 news clusters. Second set (Task 4b) is the human translations of the same clusters. All data sets are in English.

For evaluation, we used the new automatic summary evaluation metric, ROUGE[3], which was used for the first time in DUC 2004. ROUGE is a recall-based metric for fixed-length summaries which is based on n-gram co-occurrence. It reports separate scores for 1, 2, 3, and 4-gram matching between the model summaries and the summary to be evaluated. Among these different scores, unigram-based ROUGE score (ROUGE-1) has been shown to correlate with human judgements most (Lin and Hovy, 2003).

There are 10 different human judges for DUC 2003 Task 2; 8 for DUC 2004 Task 2; and 4 for DUC 2004 Task 4. However, a subset of exactly 4 different human judges produced model summaries for any given cluster. ROUGE requires a limit on the length of the summaries to be able to make a fair evaluation. To stick with the DUC 2004 specifications and to be able to compare our system with human summaries and as well as with other DUC participants, we produced 665-byte summaries for each cluster and computed ROUGE scores against human summaries.

[3] `http://www.isi.edu/~cyl/ROUGE`

3.4.2 MEAD Summarization Toolkit

We implemented our methods inside the MEAD[4] summarization system (Radev et al., 2001). MEAD is a publicly available toolkit for extractive multi-document summarization. Although it comes as a centroid-based summarization system by default, its feature set can be extended to implement any other method.

The MEAD summarizer consists of three components. During the first step, *the feature extraction*, each sentence in the input document (or cluster of documents) is converted into a feature vector using the user-defined features. Second, the feature vector is converted to a scalar value using the *combiner*. Combiner outputs a linear combination of the features by using the predefined feature weights. At the last stage known as the *re-ranker*, the scores for sentences included in related pairs are adjusted upwards or downwards based on the type of relation between the sentences in the pair. Re-ranker penalizes the sentences that are similar to the sentences already included in the summary so that a better information coverage is achieved.

Three default features that come with the MEAD distribution are Centroid, Position and Length. Position is the normalized value of the position of a sentence in the document such that the first sentence of a document gets the maximum Position value of 1, and the last sentence gets the value 0. Length is not a real feature score, but a cutoff value that ignores sentences shorter than the given threshold. Several re-rankers are implemented in MEAD, including one based on Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998) and the default re-ranker of the system based on Cross-Sentence Informational Subsumption (CSIS) (Radev, 2000). All of our experiments shown in Section 3.5 use the CSIS re-ranker.

A MEAD policy is a combination of three components: (a) the command lines

[4] `http://www.summarization.com`

```
feature LexRank LexRank.pl 0.2
Centroid 1 Position 1 LengthCutoff 9 LexRank 1
mmr-reranker-word.pl 0.5 MEAD-cosine enidf
```

Figure 3.4: A sample MEAD policy.

for all features, (b) the formula for converting the feature vector to a scalar, and (c) the command line for the re-ranker. A sample policy might be the one shown in Figure 3.4. This example indicates the three default MEAD features (Centroid, Position, LengthCutoff), and our new LexRank feature used in our experiments. Our LexRank implementation requires the cosine similarity threshold, 0.2 in the example, as an argument. Each number next to a feature name shows the relative weight of that feature (except for LengthCutoff where the number 9 indicates the threshold for selecting a sentence based on the number of the words in the sentence). The re-ranker in the example is a word-based MMR re-ranker with a cosine similarity threshold, 0.5. Finally "enidf" specifies the idf database file, which is a precomputed list of idf's for English words.

3.5 Results and Discussion

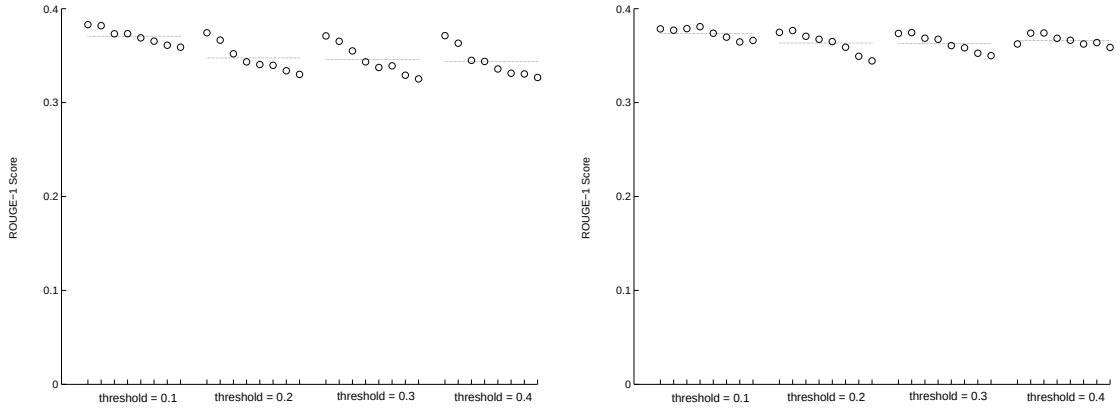
The following sections show the results of the experiments we have performed on the official DUC data sets with different implementations of similarity graph based centrality. We have implemented Degree centrality, LexRank with threshold and continuous LexRank as separate features in MEAD. All the feature values are normalized so that the sentence that has the highest value gets the score 1, and the sentence with the lowest value gets the score 0. In all of the runs, we have used Length and Position features of MEAD as supporting heuristics in addition to our centrality features. Length cutoff value is set to 9, i.e. all the sentences that have less than 9 words are discarded. The weight of the Position feature is fixed to 1 in

all runs. Other than these two heuristic features, we used each centrality feature alone without combining with other centrality methods to make a better comparison with each other. For each centrality feature we are experimenting with, we have run 8 different MEAD features by setting the weight of the corresponding feature to $0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 5.0, 10.0$, respectively.

3.5.1 Effect of Threshold on Degree and LexRank Centrality

We have demonstrated that very high thresholds may lose almost all of the information in a similarity matrix (Figure 3.3). To support our claim, we have run Degree and LexRank centrality with different thresholds for our data sets. Figure 3.5 shows the effect of threshold for Degree and LexRank centrality on DUC 2004 Task 2 data. We have experimented with four different thresholds: $0.1, 0.2, 0.3$, and 0.4. Eight different data points shown for each threshold correspond to the runs of the same feature with eight different weights as we have discussed above. The mean value of the eight different experiments is shown as a horizontal line. It is apparent in the figures that the lowest threshold, 0.1, produces the best summaries. This means that the information loss in higher thresholds is high enough to result in worse ROUGE scores. The loss in ROUGE scores as we move from threshold 0.1 to 0.2 is more significant in Degree centrality.

This effect of threshold is an indication that our new centrality methods actually *work* for extractive summarization. The higher the threshold, the less informative, or even misleading, similarity graphs we must have. On the extreme point where we have a very high threshold, we would have no edges in the graph so that Degree or LexRank centrality would be of no use.



(a) (b)

Figure 3.5: ROUGE-1 scores for (a) Degree centrality and (b) LexRank centrality with different thresholds on DUC 2004 Task 2 data.

3.5.2 Comparison of Centrality Methods

Table 3.3 shows the ROUGE scores for our experiments on DUC 2003 Task 2, DUC 2004 Task 2, DUC 2004 Task 4a, and DUC 2004 Task 4b, respectively. We show the minimum, the maximum, and the average ROUGE-1 scores for eight experiments we have run for each centrality method corresponding to eight different feature weights we have mentioned in Section 3.5. We include Degree and LexRank experiments only with threshold 0.1, the best one we have observed. We also include two baselines for each data set. The first baseline we have used is extracting random sentences from the cluster. We have performed five random runs for each data set. The results in the tables are for the median runs. The second baseline, shown as 'lead-based' in the tables, is using only the Position feature without any centrality method. This is tantamount to producing lead-based summaries, which is a widely used and very challenging baseline in the text summarization community (Brandow et al., 1995).

The top scores we have got in all data sets come from our new methods. All of our three new methods (Degree, LexRank with threshold, and continuous LexRank) perform significantly better than the baselines in all data sets. They also perform better

than centroid-based summaries except for the DUC 2003 data set where the difference between Centroid and the others is not obvious. 0.1 seems to be an appropriate threshold such that the results seem as successful as using continuous LexRank. It is also hard to say that Degree and LexRank are significantly different from each other. This is an indication that Degree may already be a good enough measure to assess the centrality of a node in the similarity graph. Considering the relatively low complexity of degree centrality, it still serves as a plausible alternative when one needs a simple implementation. Computation of Degree can be done on the fly as a side product of LexRank just before the power method is applied on the similarity graph.

To have an idea of the relative success of our methods among other summarization systems, we have compared our ROUGE scores with other participants' scores in the same DUC data sets. Table 3.4 and Table 3.5 show the official ROUGE-1 scores for top five participants and human summarizers on DUC 2003 and 2004 data, respectively. Most of the LexRank scores we got are better than the second best system in DUC 2003 and worse than the best system. Best few scores for each method are always statistically indistinguishable from the best system in the official evaluations considering the 95% confidence interval. On all three DUC 2004 data sets, we achieved a better score than the best participant in at least one of the policies we tried. On the DUC 2003 data, we achieved several scores that are between the best and the second best system.

3.5.3 Experiments on Noisy Data

The graph-based methods we have proposed consider a document cluster as a whole. The centrality of a sentence is measured by looking at the overall interaction

	2003 Task2		
	min	max	average
Centroid	0.3523	0.3713	0.3624
Degree (t=0.1)	0.3566	0.3640	0.3595
LexRank (t=0.1)	0.3610	0.3726	0.3666
Cont. LexRank	0.3594	0.3700	0.3646

baselines: random: 0.3261
lead-based: 0.3575

(a)

	2004 Task2		
	min	max	average
Centroid	0.3580	0.3767	0.3670
Degree (t=0.1)	0.3590	0.3830	0.3707
LexRank (t=0.1)	0.3646	0.3808	0.3736
Cont. LexRank	0.3617	0.3826	0.3758

baselines: random: 0.3238
lead-based: 0.3686

(b)

	2004 Task4a		
	min	max	average
Centroid	0.3768	0.3901	0.3826
Degree (t=0.1)	0.3863	0.4027	0.3928
LexRank (t=0.1)	0.3931	0.4038	0.3974
Cont. LexRank	0.3924	0.4002	0.3963

baselines: random: 0.3593
lead-based: 0.3788

(c)

	2004 Task4b		
	min	max	average
Centroid	0.3760	0.3962	0.4034
Degree (t=0.1)	0.3801	0.4147	0.4026
LexRank (t=0.1)	0.3837	0.4167	0.4052
Cont. LexRank	0.3772	0.4082	0.3966

baselines: random: 0.3734
lead-based: 0.3587

(d)

Table 3.3: ROUGE-1 scores for different MEAD policies on DUC 2003 and 2004 data.

TASK 2		
Peer Code	**ROUGE-1 Score**	**95% Confidence Interval**
C	0.4443	[0.3924,0.4963]
B	0.4425	[0.4138,0.4711]
D	0.4344	[0.3821,0.4868]
E	0.4218	[0.3871,0.4565]
A	0.4168	[0.3864,0.4472]
I	0.4055	[0.3740,0.4371]
G	0.3978	[0.3765,0.4192]
F	0.3904	[0.3596,0.4211]
J	0.3895	[0.3591,0.4199]
H	0.3869	[0.3659,0.4078]
12	0.3798	[0.3598,0.3998]
13	0.3676	[0.3507,0.3844]
16	0.3660	[0.3474,0.3846]
6	0.3607	[0.3415,0.3799]
26	0.3582	[0.3337,0.3828]

Table 3.4: Summary of official ROUGE scores for DUC 2003 Task 2. Peer codes: manual summaries [A-J] and top five system submissions.

TASK 2		
Peer Code	**ROUGE-1 Score**	**95% Confidence Interval**
H	0.4183	[0.4019,0.4346]
F	0.4125	[0.3916,0.4333]
E	0.4104	[0.3882,0.4326]
D	0.4059	[0.3870,0.4249]
B	0.4043	[0.3795,0.4291]
A	0.3933	[0.3722,0.4143]
C	0.3904	[0.3715,0.4093]
G	0.3890	[0.3679,0.4101]
65	0.3822	[0.3694,0.3951]
104	0.3744	[0.3635,0.3853]
35	0.3743	[0.3612,0.3874]
19	0.3739	[0.3608,0.3869]
124	0.3706	[0.3578,0.3835]

TASK 4		
Peer Code	**ROUGE-1 Score**	**95% Confidence Interval**
Y	0.4445	[0.4230,0.4660]
Z	0.4326	[0.4088,0.4565]
X	0.4293	[0.4068,0.4517]
W	0.4119	[0.3870,0.4368]
Task 4a		
144	0.3883	[0.3626,0.4139]
22	0.3865	[0.3635,0.4096]
107	0.3862	[0.3555,0.4168]
68	0.3816	[0.3642,0.3989]
40	0.3796	[0.3581,0.4011]
Task 4b		
23	0.4158	[0.3933,0.4382]
84	0.4101	[0.3854,0.4348]
145	0.4060	[0.3678,0.4442]
108	0.4006	[0.3700,0.4312]
69	0.3984	[0.3744,0.4225]

Table 3.5: Summary of official ROUGE scores for DUC 2004 Tasks 2 and 4. Peer codes: manual summaries [A-Z] and top five system submissions. Systems numbered 144 and 145 are University of Michigan's submission. 144 uses LexRank in combination with Centroid whereas 145 uses Centroid alone.

	2003 Task2		
	min	max	average
Centroid	0.3502	0.3689	0.3617
Degree (t=0.1)	0.3501	0.3650	0.3573
LexRank (t=0.1)	0.3493	0.3677	0.3603
Cont. LexRank	0.3564	0.3653	0.3621

baselines: random: 0.2952
lead-based: 0.3246

(a)

	2004 Task2		
	min	max	average
Centroid	0.3563	0.3732	0.3630
Degree (t=0.1)	0.3495	0.3762	0.3622
LexRank (t=0.1)	0.3512	0.3760	0.3663
Cont. LexRank	0.3465	0.3808	0.3686

baselines: random: 0.3078
lead-based: 0.3418

(b)

	2004 Task4a		
	min	max	average
Centroid	0.3706	0.3898	0.3761
Degree (t=0.1)	0.3874	0.3943	0.3906
LexRank (t=0.1)	0.3883	0.3992	0.3928
Cont. LexRank	0.3889	0.3931	0.3908

baselines: random: 0.3315
lead-based: 0.3615

(c)

	2004 Task4b		
	min	max	average
Centroid	0.3754	0.3942	0.3906
Degree (t=0.1)	0.3801	0.4090	0.3963
LexRank (t=0.1)	0.3710	0.4022	0.3911
Cont. LexRank	0.3700	0.4012	0.3905

baselines: random: 0.3391
lead-based: 0.3430

(d)

Table 3.6: ROUGE-1 scores for different MEAD policies on 17% noisy DUC 2003 and 2004 data.

of the sentence within the cluster rather than the local value of the sentence in its document. This is especially critical in generic summarization where the information unrelated to the main theme of the cluster should be excluded from the summary. DUC data sets are perfectly clustered into related documents by human assessors. To observe the behavior of our methods on noisy data, we have added 2 random documents in each cluster taken from a different cluster. Since originally each cluster contains 10 documents, this means a 2/12 (17%) noise on the data sets.

The results on the noisy data are given in Table 3.6. The picture looks similar to Table 3.3 except for lead-based and random baselines are more significantly affected by the noise. The performance loss is quite small on our graph-based centrality methods. A surprising point is that centroid-based summarization also gives good results although still worse than the others most of the time. This suggests that 17% noise on the data is not enough to make significant changes on the centroid of a cluster.

3.6 Related Work

There have been attempts for using graph-based ranking methods in natural language applications before. Salton et al. (1997) made one of the first attempts of using degree centrality in single document text summarization. In the summarization approach of Salton et al., degree scores are used to extract the important paragraphs of a text.

(Moens et al., 1999) use cosine similarity between the sentences to cluster a text into different topical regions. A predefined cosine threshold is used to cluster paragraphs around seed paragraphs (called *medoids*). Seed paragraphs are determined by maximizing the total similarity between the seed and the other paragraphs in a

cluster. The seed paragraphs are then considered as the representative descriptions of the corresponding subtopics, and included in the summary.

(Zha, 2002) defines a bipartite graph of the set of terms and the set of sentences. There is an edge from a term t to a sentence s if t occurs in s. (Zha, 2002) argues that the terms that appear in many sentences with high salience scores should have high salience scores, and the sentences that contain many terms with high salience scores should also have high salience scores. This *mutual reinforcement principal* reduces to a solution for the singular vectors of the adjacency matrix of the bipartite graph.

The work presented in this chapter started with the implementation of LexRank with threshold on unweighted graphs. This implementation was first used in the DUC 2004 evaluations which was run in February 2004 and presented in May 2004 (Erkan and Radev, 2004c). After the DUC evaluations, a more detailed analysis and more careful implementation of the method was presented together with a comparison against degree centrality and centroid-based summarization (Erkan and Radev, 2004a). Continuous LexRank on weighted graphs first appeared in the paper version of the work in this chapter submitted in July 2004 to JAIR (Erkan and Radev, 2004b). An eigenvector centrality algorithm on weighted graphs was independently proposed by (Mihalcea and Tarau, 2004) (Mihalcea and Tarau, 2004) for single-document summarization. (Mihalcea et al., 2004) later applied PageRank to another problem of natural language processing, word sense disambiguation.

Unlike our system, the studies mentioned above do not make use of any heuristic features of the sentences other than the centrality score. They do not also deal with the multi-document case. One of the main problems with multi-document summarization is the potential duplicate information coming from different documents, which is less likely to occur in single-document summaries. We try to avoid the

repeated information in the summaries by using the reranker of the MEAD system. This problem is also addressed in (Salton et al., 1997)'s work. Instead of using a reranker, they first segment the text into regions of different subtopics and then take at least one representative paragraph with the highest degree value from each region.

To determine the similarity between two sentences, we have used the cosine similarity metric that is based on word overlap and idf weighting. However, there are more advanced techniques of assessing similarity which are often used in the topical clustering of documents or sentences (Hatzivassiloglou et al., 2001; McKeown et al., 2001). The similarity computation might be improved by incorporating more features (e.g. synonym overlap, verb/argument structure overlap, stem overlap) or mechanisms (e.g. co-reference resolution, paraphrasing) into the system. These improvements are orthogonal to our model presented in this chapter and can be easily integrated into the similarity relation.

3.7 Conclusion

We have presented a new approach to define sentence salience based on graph-based centrality scoring of sentences. Constructing the similarity graph of sentences provides us with a better view of *important* sentences compared to the centroid approach, which is prone to over-generalization of the information in a document cluster. We have introduced three different methods for computing centrality in similarity graphs. The results of applying these methods on extractive summarization are quite promising. Even the simplest approach we have taken, degree centrality, is a good enough heuristic to perform better than lead-based and centroid-based summaries. In LexRank, we have tried to make use of more of the information in the graph, and got even better results in most of the cases. The accumulation of

similarity values recursively from the neighbors of each node has an impact on the centrality values of the nodes and the overall quality of the summaries we get. Lastly, we have shown that our methods are quite insensitive to noisy data that often occurs as a result of imperfect topical document clustering algorithms.

CHAPTER IV

BIASED LEXRANK FOR FOCUSED SUMMARIZATION

4.1 Introduction

Chapter III addressed the problem of *generic* summarization. However, people often prefer to see some specific information about a topic in a summary rather than a generic summary that tries to cover as much of the information from the original documents as possible. An example summarization problem from Document Understanding Conferences (DUC) 2006 is as follows:

Example IV.1.

- topic: *international adoption*
- focus: *What are the laws, problems, and issues surrounding international adoption by American families?*

Given a set of documents about a topic (e.g. "international adoption"), the systems are required to produce a summary that *focuses* on the given aspects of that topic. The LexRank method of Chapter III does not handle such an input that would require the summary to be in a focused nature. In this chapter, we describe a *topic-sensitive* extension of LexRank that can handle topic descriptions in order to produce summaries that focus on a particular aspect of a topic.

4.2 Biased LexRank

In a generalized form the LexRank equation can be written as

$$\mathrm{LR}(u) = \frac{d}{N} + (1-d) \sum_{v \in adj[u]} \frac{w(v,u)}{\sum_{z \in adj[v]} w(v,z)} \mathrm{LR}(v) \tag{4.1}$$

where $w(v,u)$ is the weight of the link from v to u. We used the cosine measure for $w(v,u)$, but any symmetric or asymmetric metric such as the generation probabilities introduced in Chapter II could be used. There is nothing in Equation 4.1 that favors certain sentences based on a topic focus: LexRank is completely *unsupervised* in the sense that it only depends on the overall structure of the graph. The first term, $\frac{d}{N}$, is introduced to make the matrix ergodic so that a solution to the equation exists. It does not have a big impact on the final ranking of the nodes since it favors all the nodes equally during the random walk. With probability d, the random walk jumps to any node with uniform probability. This suggests an alternative view of the random walk process. We can combine more than one random walks into one random walk process. Indeed, we could use a non-uniform distribution in combination with the random walk based on the weight function $w(\cdot,\cdot)$.

Suppose we have a prior belief about the ranking of the nodes in the graph. This belief might be derived from a baseline ranking method which we trust to a certain extent. For example, in the focused summarization task, we can rank the sentences by looking at their similarity to the topic description. Let $b(u)$ be the score of u based on this baseline method. We can *bias* the random walk based on $b(\cdot)$ while computing LexRank as follows:

$$\mathrm{LR}(u) = d \cdot \frac{b(u)}{\sum_{z \in C} b(z)} + (1-d) \sum_{v \in adj[u]} \frac{w(v,u)}{\sum_{z \in adj[v]} w(v,z)} \mathrm{LR}(v) \tag{4.2}$$

where C is the set of all nodes in the graph. Note that the corresponding matrix is ergodic if $b(u) > 0$ for all u's. We call Equation 4.2 *biased* or *topic-sensitive*

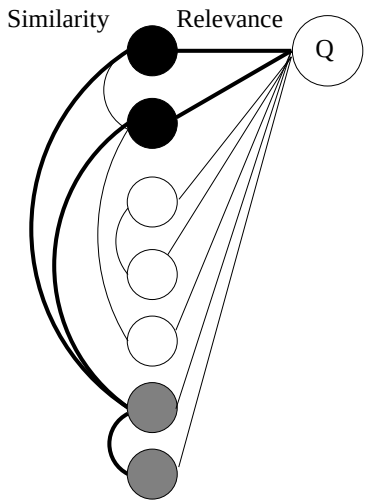


Figure 4.1: An illustration of biased LexRank. The nodes (sentences) are initially ranked by their baseline relevance to the query Q. However, the rankings of the bottom two shaded nodes are expected to be boosted up since they are similar to the top ranked nodes and each other.

LexRank since it favors certain set of sentences during the random walk based on a prior distribution. When $d = 1$, LR$(\cdot)$ ranks the nodes exactly the same as $b(\cdot)$. When $d < 1$, we have a mixture of the baseline scores and the LexRank scores derived from the *unbiased* structure of the graph. In other words, biased LexRank ranks the sentences by looking at the baseline method and the inter-sentence similarities at the same time. Figure 4.2 shows an illustration. A version of biased LexRank was successfully applied to the sentence retrieval for question answering task (Otterbacher et al., 2005).

4.3 A Question Answering Example

A problem closely related to focused summarization is question answering (QA). Actually, the only difference between QA and focused summarization is that QA deals with questions that require very specific and short answers that are usually few words long whereas summarization deals with questions that can be answered by composing a short document of few sentences such as the one in Example IV.1.

A crucial first step for most QA systems is to retrieve the sentences that potentially contain the answer to the question (Gaizauskas et al., 2004). Consider the following example.

Example IV.2.

Question: What was the **plane**'s **destination**?

1. The **plane** was **destined** for Italy's capital Rome.
2. The **plane** was in route from Locarno in Switzerland, to its **destination**, Rome, Italy.
3. The pilot was on a 20-minute flight from Locarno, Switzerland to Milan.
4. The aircraft had taken off from Locarno, Switzerland, and was heading to Milan's Linate airport.

These four sentences were taken from several news articles about the same event. There is some contradictory information among them since they come from different sources at different times. However, in a question answering scenario all of them need to be retrieved. In the absence of any useful external information, a popular sentence scoring technique in QA is to look at the words in the question. If a sentence has some words in common with the question, it is considered to be a *relevant* sentence that may contain the answer to the question. Given a sentence u and a question q, an example sentence scoring formula, taken from (Allan et al., 2003), is as follows:

$$\text{rel}(u|q) = \sum_{w \in q} \log(tf_{w,u} + 1) \times \log(tf_{w,q} + 1) \times \text{idf}_w \tag{4.3}$$

where $tf_{w,u}$ and $tf_{w,q}$ are the number of times w appears in u and q, respectively. With this technique, one is able to say that the first two sentences in Example IV.2 are somewhat related to the question since they include words from the question (shown

in boldface).[1] However, last two sentences are also clearly related to the question and actually contain the answer. An interesting observation is that sentences 3 and 4 have some words in common with sentences 1 and 2. Knowing that sentences 1 and 2 are relevant or important, it is easy to infer that sentences 3 and 4 should also be relevant only by looking at the inter-sentence similarities between them. Hence, a suitable choice for the biased LexRank formula looks like this:

$$\text{LR}(u|q) = d \cdot \frac{\text{rel}(u|q)}{\sum_{z \in C} \text{rel}(z|q)} + (1-d) \sum_{v \in adj[u]} \frac{w(v,u)}{\sum_{z \in adj[v]} w(v,z)} \text{LR}(v|q) \tag{4.4}$$

The random walk interpretation of this formula is as follows: With probability d, the random walk visits a sentence with a probability proportional to its relevance to the question, with probability $(1-d)$ the random walk chooses a sentence that is a neighbor of the current sentence with a probability proportional to the link weight in the graph. So, the random walk is biased towards the *neighborhoods* of the highly relevant sentences in the graph. In (Otterbacher et al., 2005), we applied this formula to the sentence retrieval problem in question answering, where we achieved significant improvements over the baseline scoring system in terms of sentence ranking.

4.4 Application to Focused Summarization

DUC summarization evaluations in 2005 and 2006 included a focused summarization task. Given a topic and a set of 25 relevant documents, the participants were required "to synthesize a fluent, well-organized 250-word summary of the documents that answers the question(s) in the topic statement". An example topic statement and related questions are in Example IV.1. In this section, we explain how we formulated the summarization tasks of DUC 2005 and 2006 based on the biased LexRank technique of Section 4.2.

[1] *was* and *the* are considered to be stop words, so they are not included in the word matching process.

4.4.1 Using Generation Probabilities as Link Weights

As explained in Chapter III, we used the cosine measure for the edge weights of the sentence similarity graphs in DUC 2004. Other than replacing LexRank with biased LexRank, one fundamental change in our approach for DUC 2005 and 2006 is to replace cosine with language model based similarity functions (Section 2.3). Our motivation comes from the work of (Kurland and Lee, 2005) which proposes a document retrieval method that is similar to LexRank. The main differences of their approach from our original LexRank formulation are that they use documents instead of sentences, and they define the edge weight $w(u, v)$ from document u to document v as the KL divergence of the language models of u given v (Equation 2.14). They show that using KL divergence as edge weights yields better results than cosine. In this section, we adopt a similar similarity function for the edge weights.

First we recall the language modeling approach from Chapter II and adapt it to the summarization domain. Given a sentence v, we can compute a (unigram) language model from it. A straightforward way of computing this language model is the maximum likelihood estimation (MLE) (Equation 2.6) of the probabilities of the words to occur in v:

$$p_{\mathrm{ML}}(w|v) = \frac{\mathrm{tf}_{w,v}}{|v|} \tag{4.5}$$

The MLE is often not a good approximation for a language model since the words that do not occur in the text that we compute the word frequencies from get zero probability. This is an even bigger problem when we compute language models from a relatively shorter input text such as a sentence composed of few words. To account for the unseen words, we smooth the language model computed from a sentence using

the language model computed from the entire cluster:

$$p_{\text{JM}}(w|v) = (1-\lambda)p_{\text{ML}}(w|v) + \lambda p_{\text{ML}}(w|C) \tag{4.6}$$

where C is the entire document cluster. Equation 4.6 is a special instance of the more general Jelinek-Mercer smoothing method (Equation 2.9), where the constant $\lambda \in [0,1]$ is a trade-off parameter between the MLE computed from the sentence and the MLE computed from the entire cluster. ${}_{\text{JM}}p(w|v)$ is nonzero for all words that occur in the document cluster to be summarized provided that $\lambda > 0$.

We can also talk about the generation probability of a sentence given the language model computed from another sentence. For example,

$$p_{\text{gen}}(u|v) = \prod_{w \in u} p_{\text{JM}}(w|v)^{\text{tf}_{w,u}} \tag{4.7}$$

defines the generation probability of sentence u given the language model of sentence v. Since the probabilities of all words get multiplied with each other, longer sentences tend to get smaller generation probabilities. Therefore, as in Equation 2.13, we normalize the generation probability of each sentence by the sentence length to make different pairwise similarities comparable with each other:

$$p_{\text{norm}}(u|v) = \left(\prod_{w \in u} p_{\text{JM}}(w|v)^{\text{tf}_{w,u}} \right)^{\frac{1}{|u|}} \tag{4.8}$$

We use $p_{\text{norm}}(u|v)$ as the weight of the link *from* u *to* v in the graph-based representation of the cluster. Note that $p_{\text{norm}}(u|v)$ is not necessarily equal to $p_{\text{norm}}(v|u)$. The probability of a 1-step random walk (i.e. a random walk of length 1) from u to v is proportional to the (normalized) generation probability of u given the language model computed from v. The reason we are not using the value for the opposite direction ($p_{\text{norm}}(v|u)$) is that each sentence should get credit for its capacity to generate the other sentences, not to be generated by them. If a sentence has strong

incoming generation links in the graph, it is an evidence that the language model of that sentence can generate other sentences more successfully. Revisiting the random walk model of LexRank, the LexRank value of of a sentence is a measure of its accumulated *generation power*, that is, how likely it is to generate the rest of the cluster from the language model of the specific sentence in the long run. We advocate that a sentence with a high generation power is a good candidate for the summary of its cluster.

Extending the use of generation probabilities to biased LexRank for the focused summarization task is straightforward. For the baseline ranking method, we use the generation probability of the topic description from the sentences. A sentence is ranked higher if its language model can generate the topic description with a larger probability. This is analogous to the language modeling approach in information retrieval (Ponte and Croft, 1998) where the documents are ranked with respect to the generation probability of the given query from each document's language model. In our summarization method, given a topic description t, the final score for a sentence u is computed by the following biased LexRank equation:

$$\text{LR}(u|t) = d \cdot \frac{p_{\text{gen}}(t|u)}{\sum_{z \in C} p_{\text{gen}}(t|z)} + (1-d) \sum_{v \in adj[u]} \frac{p_{\text{norm}}(v|u)}{\sum_{z \in adj[v]} p_{\text{norm}}(v|z)} \text{LR}(v|t) \tag{4.9}$$

4.4.2 Experiments and Results on DUC 2005 and 2006

In DUC 2005 and 2006 summarization evaluations, the task was to summarize a set of documents based on a particular aspect of their common topic. This particular aspect was provided as a "topic description" (see Example IV.1). Other than this change, the setting was similar to DUC 2003 and 2004 evaluations: There were 50 topical clusters to be summarized for each year. Each cluster had 25 English news documents that talked about the same topic.

There are two parameters in our framework summarized by Equation 4.9 above. d is the biased jump probability in biased LexRank, which is a trade-off between the similarity of a sentence to the topic description and to other sentences in the cluster. λ is the Jelinek-Mercer smoothing parameter (Equation 4.6) that is used when computing the language model of each sentence. For both parameters, we experimented with several values in the $[0.1, 0.9]$ interval. Here we report the results for the one of the best parameter settings we got for the DUC 2005 dataset. We want to mention that we did not carry out an extensive parameter tuning. Our purpose is to show that the biased LexRank method is effective even with poor parameter tuning. To further support this claim, we did not perform any parameter tuning for the DUC 2006 dataset at all, directly using the same parameter values from the DUC 2005 experiments. Overall, $d \approx 0.7$ and $\lambda \approx 0.6$ performed well. Note that 0.7 is a relatively large value for d considering that we set it to 0.15 in the generic summarization experiments. However, large values for d makes perfect sense for focused summarization since we would certainly want to give more weight to a sentence's similarity to the topic description than its similarities to other sentences. For small values of d, the summaries would be more generic rather than based on the topic description.

Another change we made in the similarity graphs for focused summarization was to connect each node (sentence) to k other nodes that are most similar to it rather than connecting to all the other nodes in the graph. We observed that this improves not only the running time of the algorithm but also the quality of the resultant summaries. One reason for this may be that small similarity values actually indicate "dissimilarity" among sentences. Accumulating scores from dissimilar neighbors of a node is not intuitive no matter how small the similarity values are. In our experi-

ments, we considered only the top 20 most similar neighbors for each sentence, that is, each node in the similarity graphs has exactly 20 outgoing links. Note that the number of incoming links for each node may vary depending on how similar a node is to the rest of the nodes in the graph. Indeed, a good summary sentence would typically have more incoming links than outgoing links, which is an indication that its language model can better generate the rest of the sentences in the graph.

We did not use any features other than the biased LexRank values in our experiments. To construct the final summaries, we ranked the sentences based on their biased LexRank scores. We put the ranked sentences in a summary one by one until the summary exceeded 250 words which was the limit in the DUC evaluations. When putting the sentences in the ranked order into the summary, we ignored any sentence which has a *cosine* similarity of larger than 0.5 to any of the sentences that were ranked above it. This simple reranking scheme ensures that the summaries we get cover as much information as possible within the length limit.

For the evaluation of our summaries, we used the official ROUGE metrics of DUC 2005 and 2006, i.e. ROUGE-2 and ROUGE-SU4. ROUGE-2 looks at the bigram overlap between the system summary and the manual summaries created by humans. ROUGE-SU4 does the same except with the relaxation of allowing as many as four words between the two words of a bigram. Tables 4.1 and 4.2 show the results for DUC 2005 and 2006, respectively. In both datasets, biased LexRank performs comparable to the human summarizers. In some cases, its performance is not significantly different from certain human summarizers considering the 95% confidence intervals. For comparison, we also include the top ranked system's score each year. LexRank achieves almost the same scores or better.

4.5 Conclusion

In this chapter, we have presented an extension to the LexRank summarization framework that enables us to produce summaries of documents based on a particular aspect of a topic given as a query. The results show that LexRank is still very competitive for this problem. We have also introduced the use of language modeling based generation probabilities as the text similarity function to construct the similarity graph instead of the *cosine* function of Chapter III.

	ROUGE-2		ROUGE-SU4	
A	0.11711	[0.10578,0.12865]	0.17560	[0.16595,0.18552]
B	0.10020	[0.08643,0.11533]	0.16102	[0.14997,0.17283]
C	0.11785	[0.10469,0.13062]	0.17798	[0.16587,0.18931]
D	0.10035	[0.08880,0.11336]	0.16155	[0.15120,0.17342]
E	0.10508	[0.09160,0.11959]	0.15922	[0.14712,0.17163]
F	0.09977	[0.08877,0.11138]	0.15822	[0.14781,0.16919]
G	0.09683	[0.08546,0.10833]	0.15954	[0.15010,0.16905]
H	0.08811	[0.07682,0.10055]	0.14798	[0.13795,0.16023]
I	0.09888	[0.08467,0.11642]	0.16116	[0.14652,0.17897]
J	0.09968	[0.08960,0.11133]	0.16058	[0.15126,0.17180]
LR	0.07531	[0.07203,0.07858]	0.13630	[0.13274,0.13969]
Best	0.07440	[0.07169,0.07736]	0.13458	[0.13173,0.13749]

Table 4.1: ROUGE-2 and ROUGE-SU4 score comparison against the human summarizers in DUC 2005 along with their 95% confidence intervals. A-J: human summarizers; LR: biased LexRank; 409 Best: best system in the original evaluations.

	ROUGE-2		ROUGE-SU4	
A	0.10361	[0.09260,0.11617]	0.16829	[0.16042,0.17730]
B	0.11788	[0.10501,0.13351]	0.17665	[0.16356,0.19080]
C	0.13260	[0.11596,0.15197]	0.18385	[0.17012,0.19878]
D	0.12380	[0.10751,0.14003]	0.17814	[0.16527,0.19094]
E	0.10365	[0.08935,0.11926]	0.16298	[0.15012,0.17606]
F	0.10893	[0.09310,0.12780]	0.16043	[0.14518,0.17771]
G	0.11324	[0.10195,0.12366]	0.17121	[0.16301,0.17952]
H	0.10777	[0.09833,0.11746]	0.16665	[0.15627,0.17668]
I	0.10634	[0.09632,0.11628]	0.16843	[0.15828,0.17851]
J	0.10717	[0.09293,0.12460]	0.16934	[0.15716,0.18319]
LR	0.09232	[0.08806,0.09684]	0.15010	[0.14598,0.15417]
Best	0.09505	[0.09093,0.09914]	0.15464	[0.15061,0.15837]

Table 4.2: ROUGE-2 and ROUGE-SU4 score comparison against the human summarizers in DUC 2006 along with their 95% confidence intervals. A-J: human summarizers; LR: biased LexRank; Best: best system in the original evaluations.

CHAPTER V

LANGUAGE MODEL BASED DOCUMENT CLUSTERING USING RANDOM WALKS

5.1 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 (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.

5.2 Generation Probabilities as Document Vectors

In this section, we present our document vector representation model based on the pairwise language model-based probabilities between the documents.

5.2.1 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 (Equation 2.6). For each term w that occurs in a document D, the ML estimation of w with respect to D is defined as

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

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:

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

where $p_{\mathrm{ML}}(w|Corpus)$ 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) that we introduced in Section 2.3 (Equation 2.10):

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

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:

$$p(S|D) = \prod_{w \in S} p(w|D) \tag{5.1}$$

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 5.1, 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:

$$p_{\text{norm}}(S|D) = p(S|D)^{\frac{1}{|S|}} \tag{5.2}$$

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

5.2.2 Using Generation Probabilities as Document Representations

Equation 5.2 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* $\mathbf{g}_{d_i} = (g_{d_i1}, g_{d_i2}, \ldots, g_{d_in})$ for each document d_i where

$$g_{d_ij} = \begin{cases} 0 & \text{if } i = j, \\ p_{\text{norm}}(d_i|d_j) & \text{otherwise} \end{cases} \tag{5.3}$$

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 $p_{\text{norm}}(d_i|d_j)$. 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 5.2 with a smoothed language model is never zero. This creates two potential problems if we want to use the vector of Equation 5.3 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 $\mathbf{g}_{d_i}^c$, 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.

5.2.3 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 V.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 V.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 (Page et al., 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 V.3. The *t-step generation probability* of document d_i from the language model of d_j:

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

$\mathbf{gen}^t_{d_i} = (\text{gen}^t(d_i|d_1), \text{gen}^t(d_i|d_2), \ldots, \text{gen}^t(d_i|d_n))$ 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.

5.3 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, 2004b; 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 (2004b) 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 chapter 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 5.4.

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).

5.4 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.

5.4.1 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:

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

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

We computed flattened generation probabilities (Equation 5.2) for all ordered pairs of documents in a corpus, and then constructed the corresponding generation graph (Definition V.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 5.2.2.

- 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.

5.4.2 Experiments with k-means

We use k-means, one of the most popular clustering algorithms, in our first set of experiments.

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.

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

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

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 5.1 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 5.1: The corpora used in the k-means experiments.

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

[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 5.4.1

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

as:

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

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

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

One disadvantage of k-means is that its performance is very much 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.

Tables 5.2 and 5.3 show 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

Corpus	k	Accuracy (×100)			
		$tf \cdot idf$	$\mathbf{gen}^1$	$\mathbf{gen}^2$	$\mathbf{gen}^3$
classic3	3	95.76±1.28	97.92±0.69	98.70±0.19	**98.79±0.03**
4news-1	4	72.32±2.80	82.76±1.55	85.86±1.30	**86.58±1.23**
4news-2	4	63.42±2.83	77.33±2.14	80.66±1.92	**81.29±1.80**
4news-3	4	62.35±2.51	74.96±3.35	78.60±3.67	**78.85±3.75**
4news-4	4	73.14±2.05	81.36±1.54	83.54±1.50	**84.22±1.49**
4news-5	4	69.43±3.68	87.06±2.33	**90.07±1.66**	90.06±1.47
5news-1	5	59.00±2.43	73.91±2.30	76.35±2.86	**76.75±2.58**
5news-2	5	55.59±2.88	69.32±2.17	72.75±1.74	**73.05±1.82**
5news-3	5	71.07±2.39	83.78±2.16	86.14±2.15	**86.28±2.22**
5news-4	5	70.06±2.64	80.20±1.84	82.61±1.69	**82.66±1.78**
bbc	5	80.73±2.80	87.46±2.63	89.88±2.57	**90.80±2.26**
bbcsport	5	79.25±2.87	94.25±0.89	95.44±0.42	**95.56±0.31**
10news-1	10	50.11±2.30	66.20±2.12	69.18±1.73	**69.76±1.61**
10news-2	10	54.12±1.76	70.01±2.00	73.41±1.78	**74.14±1.75**
20news	20	41.46±1.03	55.75±1.25	58.97±1.29	**60.10±1.12**

Table 5.2: k-means accuracy using different vector representations (average of 30 runs ±95% confidence interval).

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 Tables 5.2 and 5.3 is that even $\mathbf{gen}^1$ vectors perform better than the $tf \cdot idf$ 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]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.

Corpus	k	Normalized Mutual Information (×100)			
		$tf \cdot idf$	$\mathbf{gen}^1$	$\mathbf{gen}^2$	$\mathbf{gen}^3$
classic3	3	84.69±2.50	91.16±1.90	93.39±0.59	**93.64±0.12**
4news-1	4	49.85±3.39	64.85±1.56	69.72±0.88	**70.73±0.89**
4news-2	4	34.02±2.97	54.55±1.90	59.50±1.45	**60.46±1.31**
4news-3	4	33.74±2.65	51.94±3.43	58.15±3.08	**59.24±2.95**
4news-4	4	54.24±2.74	66.47±1.24	69.78±0.90	**70.41±0.83**
4news-5	4	42.58±3.90	66.49±3.27	71.95±2.45	**71.96±2.15**
5news-1	5	36.53±2.94	56.84±3.19	60.74±2.87	**61.40±2.54**
5news-2	5	30.94±2.45	48.00±2.18	52.77±1.52	**53.75±1.30**
5news-3	5	49.65±2.64	67.09±2.12	71.13±1.87	**71.70±1.67**
5news-4	5	50.04±2.82	63.69±1.73	66.89±1.32	**67.49±1.14**
bbc	5	63.34±3.23	74.19±2.93	78.20±2.63	**79.39±2.30**
bbcsport	5	63.94±3.27	84.59±1.34	86.42±0.71	**86.54±0.58**
10news-1	10	39.98±1.99	55.21±1.67	58.86±1.15	**59.70±0.96**
10news-2	10	42.44±1.55	60.64±1.40	65.20±1.11	**66.41±1.02**
20news	20	38.28±0.73	52.44±0.74	56.34±0.71	**57.37±0.60**

Table 5.3: k-means normalized mutual information using different vector representations (average of 30 runs ±95% confidence interval).

5.4.3 Experiments with Hierarchical Clustering

Hierarchical clustering is another popular set of clustering algorithms which this section is based on.

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.

	TDT2				Reuters-21578			
Algorithm	$tf{\cdot}idf$	$\mathbf{gen}^1$	$\mathbf{gen}^2$	$\mathbf{gen}^3$	$tf{\cdot}idf$	$\mathbf{gen}^1$	$\mathbf{gen}^2$	$\mathbf{gen}^3$
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 5.4: Performances (F-measure $\times 100$) of different vector representations using hierarchical algorithms on two corpora.

Data

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

Table 5.5: 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 at 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 5.5.

[5] `http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html`
[6] `http://www.nist.gov/speech/tests/tdt/tdt98/index.htm`

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

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

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 5.4 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.

5.5 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. This is a strong indication that short random walks we have considered in this chapter yields better similarity values among the documents by taking into account the indirect similarity relationships among them.

CHAPTER VI

IMPROVED NEAREST NEIGHBOR METHODS FOR TEXT CLASSIFICATION

6.1 Introduction

Text classification is the task of assigning documents to the most likely category or categories. Classifying news articles into categories such as sports, politics, arts, etc. is a typical example. Formally, given a set of documents, $\mathcal{D}$, and a set of categories, $\mathcal{C}$, we assume that there is a *true* function $\hat{f} : \mathcal{D} \times \mathcal{C} \rightarrow \{0, 1\}$ that assigns documents to categories such that a document $d \in \mathcal{D}$ belongs to a category $c \in \mathcal{C}$ if and only if $\hat{f}(d, c) = 1$. Text classification tries to find (learn) an approximation f to the $\hat{f}$ function. If $\hat{f}$ assigns each document to exactly one category, then the classification task is *single-label*, otherwise it is *multi-label* classification. For example, a news article can be classified into both "middle east" and "economy" categories. Depending on the learning algorithm used, we might have f take any real value in the $[0, 1]$ interval, which is called *soft* classification, or try to predict $\hat{f}$ directly by forcing f to take only binary values. Soft classification is usually followed by a thresholding operation on the f values to perform the final category assignments.

Text classification has been one of the most popular problems in information retrieval and machine learning. The vast number of its potential applications, the

availability of huge (but mostly unlabeled) data especially on the Web and the high dimensionality of its feature space make it a particularly interesting testbed for machine learning methods in general. It would not be an exaggeration to say that nearly all known machine learning methods, from neural networks (Schütze et al., 1995) to Support Vector Machines (Joachims, 1998), have been applied to text classification where applicable (Sebastiani, 2002). Text classification is sometimes the first domain to test a new machine learning algorithm.

Among many approaches to text classification, here we primarily focus on the *nearest neighbor* (NN) based approaches. Nearest neighbor approaches try to classify a document to a category by looking at the categories of the documents that are most similar to it. For this reason, they are also known as *case-based* learning algorithms. Since they are based on similarity, it is tempting to think that they may perform well in text classification given the success of other similarity and graph based methods on other problems we presented in previous chapters. It is well known that NN based approaches suffer from irrelevant features (Kohavi et al., 1997). When two instances in a learning problem have too many irrelevant features in common, the similarity metric yields a large value while these two instances may not be similar or belong to the same category in reality. However, feature relevance is a well-studied problem for documents in information retrieval. It is precisely the motivation behind the techniques such as stop word removal, $tf \cdot idf$, language models, and the similarity functions that make use of these techniques. Therefore the similarity functions in the text domain that we have presented in Chapter II are arguably better approximations to *real* similarities and suffer less from irrelevant features compared to the generic similarity functions used in machine learning (e.g. simple Euclidean distance without feature selection). Furthermore, we have seen in previous chapters

that by using graph-based techniques such as random walks, we can improve these similarity metrics even further. In Chapter V, NN based techniques combined with random walks proved to be very successful in document clustering, which is simply nothing but the unsupervised version of text classification.

NN based methods are also intuitive in the text domain. Knowing that a document belongs to a certain category, we can usually classify another document to the same category if it is very similar to the first one even though we might not have an idea of what the category is or we might not have seen any other negative or positive examples. The list of words in a document is often a good description of its category. For all these reasons, NN based methods seem to be promising for text classification especially if we can reuse ideas from previous chapters to improve the similarity notion. Indeed, even the simplest NN methods such as k-nearest neighbor algorithm (kNN) have been shown to perform surprisingly well in text classification (Joachims, 1998; Yang and Liu, 1999). Despite this success, little effort has been made to improve kNN's performance further. Instead, it has been used as a popular choice of a baseline in many studies to compare it against the proposed method.

There are two critical decisions in a NN based learning algorithm. The first is how the "nearest" neighbors of an instance (document) are determined. The second is how the category of a document is determined by looking at its nearest neighbors. In this chapter, we present several alternatives to these two problems. To find the nearest neighbors of a document, we make use of the relevance measures recently popularized in language modeling based document retrieval research (Lafferty and Zhai, 2001). We show that this improves the classification results compared to the classical approach based on the cosine similarity measure.

While combining the labels of the neighbors of a document, we also consider a semi-

supervised version of the kNN algorithm where we look at the unlabeled neighbors of a document together with its labeled neighbors. This algorithm turns out to be equivalent to the semi-supervised learning method based on harmonic functions (Zhu et al., 2003) and greatly improves the classification results when there are limited number of training documents. The use of semi-supervised algorithms with limited training data is essential in text classification. This is not only due to the numerous theoretical and empirical justifications of using unlabeled data and the cost of getting training data in the machine learning literature, but also due to the simple fact that some text classification problems inherently have limited training data. For example, consider the problem of classifying one's emails into personal folders (Kenrick Mock, 2001; Bekkerman et al., 2004). The training dataset for this problem is the set of emails that already have been received and classified. There is no way of enlarging this set unless the specific person receives and classifies more emails. A similar problem is classifying the Web pages into a user's bookmarks directories.

This chapter is organized as follows. In Section 6.2, we present the popular Naive Bayes text classification algorithm. Naive Bayes has important connections to language models which will be used repeatedly in the subsequent chapters. Section 6.3 presents several versions of the k-nearest neighbor (kNN) algorithm, alternative similarity measures and a semi-supervised version of kNN. We explain our experiments and results of comparing several nearest neighbor methods against each other as well as against Naive Bayes and Support Vector Machines in Section 6.4.

6.2 Naive Bayes

We start with Naive Bayes (NB), one of the simplest yet effective generative models for text classification. NB classification has interesting properties which are im-

portant in developing our ideas in the sections that follow. Using Bayes rule, the probability that a document d of length l belongs to the category c is determined by the following equation:

$$p(c|d) = \frac{p(d|c,l) \cdot p(c|l)}{\sum_{c'} p(d|c',l) \cdot p(c',l)} \tag{6.1}$$

where $p(d|c,l)$ is the probability of observing d given the class c and its length l. It is often assumed that the probabilities in Equation 6.1 do not depend on document length, so l is dropped from the equation. Therefore, $p(d|c,l)$ becomes $p(d|c)$, and $p(c|l)$ becomes $p(c)$. $p(c)$ is the prior probability of the category c and can be computed from the training data:

$$p(c) = \frac{|c|}{\sum_{c'} |c'|} \tag{6.2}$$

where $|c|$ is the number of documents that belong to category c in the training data.

In the most popular version of the Naive Bayes model, the words in a document are assumed to be drawn from an underlying multinomial distribution independently of each other. Since all document lengths are assumed to be equally likely, we have

$$p(d|c) \approx \prod_{w \in d} p(w|c)^{\mathrm{tf}_{w,d}} \tag{6.3}$$

where $\mathrm{tf}_{w,d}$ is the number of times word w occurs in d. $p(w|c)$ can be estimated from the relative frequencies of the words in the documents that belong to category c in the training set. However, to avoid zero probability for unseen words in a category, a smoothing method needs to be used. As in Chapter V, we choose to use Bayesian smoothing with a Dirichlet prior (Zhai and Lafferty, 2004; Liu and Croft, 2004):

$$p(w|c) = \frac{\mathrm{tf}_{w,c} + \mu \cdot p(w|Corpus)}{\sum_{w' \in c} \mathrm{tf}_{w,c} + \mu} \tag{6.4}$$

where μ is the Dirichlet smoothing parameter and $p(w|Corpus)$ is the probability of the word w in the entire training set of documents in all categories:

$$p(w|Corpus) = \frac{\mathrm{tf}_{w,Corpus}}{\sum_{w' \in Corpus} \mathrm{tf}_{w',Corpus}} \tag{6.5}$$

The denominator in Equation 6.1 is the same for all categories, thus it does not affect the category assignment decision for a given document. Therefore, the final category assignment for an unlabeled document is determined by:[1]

$$y_{\mathrm{NB}}(d) = \operatorname*{argmax}_c \, p(c) \cdot \prod_{w \in d} p(w|c)^{\mathrm{tf}_{w,d}} \tag{6.6}$$

6.3 Similarity-based Approaches

In this section we present the existing similarity-based nearest neighbor approaches and our extensions to them.

6.3.1 k-Nearest Neighbor

A different class of approaches includes nearest neighbor methods where a document is categorized by only looking at the training documents that are most similar to it. Let U be the set of unlabeled documents, and L be the set of labeled documents. Given a document $d \in U$, let $NN_k^L(d)$ be the set of the top k documents in L that are most similar to d with respect to some similarity measure. In the simplest version of the k-nearest neighbor (kNN) algorithm, d is assigned to the category that the majority of the documents in $NN_k^L(d)$ belong to:

$$y_{\mathrm{voting\_kNN}}(d) = \operatorname*{argmax}_c \sum_{\substack{d' \in NN_k^L(d) \\ y(d')=c}} 1 \tag{6.7}$$

[1] We will be using $y(d)$ to denote the category of the document d throughout this chapter. When d is unlabeled, $y(d)$ denotes the category assigned by the learning algorithm. When d is labeled, $y(d)$ is its actual label in the training data.

We call this method *voting kNN*.

Another version of kNN, which we will call *weighted kNN* takes the magnitude of the similarity values into account. The weighted kNN decision rule can be written as:

$$y_{\text{weighted\_kNN}}(d) = \operatorname*{argmax}_{c} \sum_{\substack{d' \in NN_k^L(d) \\ y(d')=c}} sim(d, d') \tag{6.8}$$

where $sim(d, d')$ is the similarity between d and d'.

6.3.2 Similarity Functions

By far the single most popular similarity function used in text classification by kNN is the well-known *cosine* measure defined on the document vectors in the tf or $tf{\cdot}idf$ weighted term space. One is tempted to ask whether using any other similarity function in kNN would improve the classification results.

One inspiration for alternative similarity measures comes from recent research in information retrieval. The language modeling (LM) approach to document retrieval, first introduced by Ponte and Croft (Ponte and Croft, 1998), has proven to be more effective than the traditional cosine retrieval model. In its most basic form, the LM retrieval model assigns a probability to a given query q for each document d in the collection:

$$p_{\text{gen}}(q|d) = \prod_{w \in q} p(w|d)^{\text{tf}_{w,q}} \tag{6.9}$$

The above equation looks exactly like Equation 6.3. There we computed the probability of a document given the term distribution of a set of documents (category) while in Equation 6.9 the same probability is computed for a query given the term distribution of a single document. As we introduced in Chapter I (see Equation 2.7),

$p(q|d)$ is called the *generation probability* of q given the *language model* $p(\cdot|d)$ of d in LM terms.

The generation probability is a measure of how related (or similar) the query is to a document and has been shown to perform better in document retrieval than the $tf \cdot idf$ cosine based retrieval model (Ponte and Croft, 1998). We can turn it into a document similarity measure by replacing q in Equation 6.9 with a document so that $p_{\text{LM}}(d|d')$ becomes the similarity of d' to d.

Using generation probabilities in weighted kNN has connections to ensemble methods in machine learning which aim to build a classifier by taking weighted votes from a "committee" of classifiers (Dietterich, 2000). Weighted kNN in combination with the sim_{LM} measure can be seen as a special case of *Bayesian voting*, one of the simplest ensemble methods, where we construct a Naive Bayes classifier from individual documents (nearest neighbors) and take their weighted votes (Equation 6.8) to make the final classification decision. Previous research has shown that an ensemble classifier is often more accurate than any of the classifiers that it is derived from (Opitz and Maclin, 1999; Dietterich, 2000).

There is one problem with using language model probabilities in nearest neighbor classification. Since all the word probabilities are multiplied to find the generation probability of a given document, small differences among different language models may result in huge differences in terms of the ratio of these different generation probabilities to each other. Indeed, Naive Bayes is known to produce overconfident probabilities for this reason. Rennie (Rennie, 2001) empirically showed that the (normalized) Naive Bayes probability for the top category is close to 1.0 for most of the documents in a classification dataset. In our experiments, we have tried to see if this property holds for pairwise document-document generation probabilities as well.

For each document d in the two datasets we use in this chapter (see Section 6.4.1), we have computed $p_{\text{LM}}(d|d')$ for all other documents d' in the dataset to find the 30 nearest neighbors of d. For 17,879 documents out of a total of 18,828 documents in the 20 Newsgroups dataset, the top nearest neighbor has a similarity value larger than the sum of the similarity values of the remaining 29 neighbors (Table 6.3.2). This is not the case for any of the documents if we use cosine as the similarity measure. This essentially means that we do not gain anything by taking weighted votes from the neighbors. In other words, using weighted kNN for these 17,879 documents is effectively the same as assigning each of them to the category that its top nearest neighbor belongs to.

Another similarity measure used in information retrieval is based on the Kullback-Leibler (KL) divergence (Lafferty and Zhai, 2001) which we introduced in Chapter II. Following Equation 2.16, the KL divergence-based similarity between a query q and a document d is computed as follows:

$$\begin{aligned} \text{sim}_{\text{KL}}(d,q) &= \exp(-\text{KL}(q \parallel d)) \\ &= \prod_{w \in q} p(w|d)^{\frac{\text{tf}_{w,q}}{|q|}} \cdot \prod_{w \in q} \left(\frac{1}{p(w|q)}\right)^{p(w|q)} \end{aligned} \tag{6.10}$$

We can ignore the second factor in the product as it does not depend on the document, so it is a constant scaling factor for a given query. The first factor in the multiplication looks very similar to the p_{LM} value in Equation 6.9. The only difference is the $1/|q|$ factor in the exponent, where $|q|$ is the number of words in q. Thus, instead of multiplying the probabilities of all the words in q in Equation 6.9, we take the geometric mean of these probabilities in Equation 6.10. Note that for a given query, Equation 6.9 and 6.10 will produce exactly the same ranking of documents (the second term and the $1/|q|$ factor in Equation 6.10 depend only on the query and

do not change the ranking.) Therefore, there is no practical difference between the two equations from the document retrieval perspective. This also means that if we use Equation 6.10 as a document similarity measure by replacing q with a document (as we have done for p_{LM}), the set of k nearest neighbors of a given document will be exactly the same as the set of k nearest neighbors that we get from p_{LM}. Thus, the voting kNN classification will also produce the same results for both similarity measures. However, the similarity values computed by $\exp(-\text{KL})$ will be "tighter" because of the geometric mean operation involved. This is obvious in Table 6.3.2 where we see that although the set of 30 nearest neighbors for a document is the same with respect to both p_{LM} and $\exp(-\text{KL})$, similarity values computed using KL divergence are closer to each other. The $\exp(-\text{KL}(\cdot|\cdot))$ function has been used as a document-document similarity in recent research and has helped improve the quality of document retrieval (Kurland and Lee, 2005) and document clustering (Erkan, 2006a).

Dataset	*cosine*	p_{LM}	$\exp(-\text{KL})$
20 News (18828)	0	17879	8
Reuters (10789)	0	8985	0

Table 6.1: The number of documents for which the similarity of their top nearest neighbor with respect to different similarity measures is larger than the sum of the similarities of the next 29 nearest neighbors.

6.3.3 Semi-supervised kNN and Harmonic Functions

Consider a binary classification problem where each document in the training set is labeled as 0 or 1. For this special case, we can write the weighted kNN equation as follows:

$$y(d) = \frac{\sum_{d' \in N_k^L(d)} sim(d, d') y(d')}{\sum_{d' \in N_k^L(d)} sim(d, d')} \tag{6.11}$$

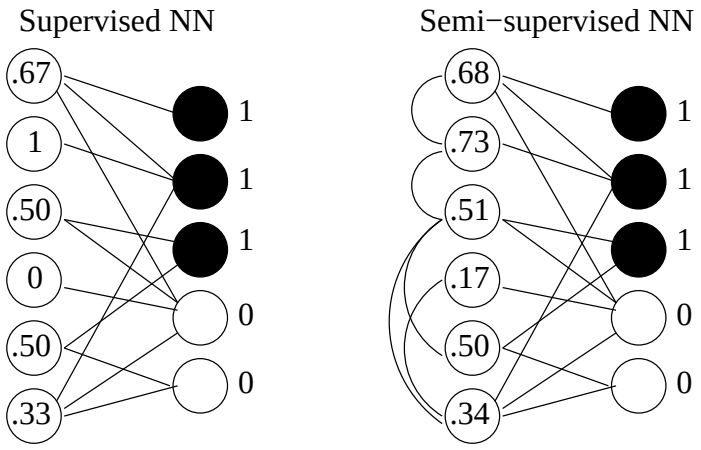


Figure 6.1: An illustration of the difference between supervised and semi-supervised nearest neighbor algorithms. On both graphs, the nodes on the right are labeled, and the nodes on the left are unlabeled. The edge weights are constant in this example. In the semi-supervised version, the values of the unlabeled neighbors are also taken into account when computing the final values.

In other words $y(d)$ is set to the weighted average of its neighbors' labels $y(d') \in \{0, 1\}$. Note that $y(d)$ can take any real value in the $[0, 1]$ interval. If we classify each unlabeled document d that has $y(d) < 0.5$ as negative (class 0), and $y(d) > 0.5$ as positive (class 1), Equation 6.11 functions exactly the same as the weighted kNN classification of Equation 6.8 for binary classification.

Equation 6.11 suggests a generalized semi-supervised version of the same algorithm by incorporating unlabeled instances as neighbors as well:

$$y(d) = \frac{\sum_{d' \in N_k^{L \cup U}(d)} sim(d, d') y(d')}{\sum_{d' \in N_k^{L \cup U}(d)} sim(d, d')} \tag{6.12}$$

Unlike Equation 6.11, the unlabeled documents are also considered in Equation 6.12 when finding the nearest neighbors. We can visualize this as a graph, where each data instance (labeled or unlabeled) is a node that is connected to its k nearest neighbor nodes. The value of $y(\cdot)$ is set to 0 or 1 for labeled nodes depending on their class. For each unlabeled node x, $y(x)$ is equal to the average of the $y(\cdot)$ values of its neighbors. Since the labels of the unlabeled documents appear on both sides of Equation 6.12, it

is not obvious whether a solution for $y(d)$ exists. Such a function that is set to fixed values ($\{0, 1\}$ in this case) for labeled nodes, and satisfies the averaging property on all unlabeled nodes is called a *harmonic* function. Fortunately, harmonic functions on a graph are known to have a unique solution (Doyle and Snell, 1984). Harmonic functions were first introduced as a semi-supervised learning method by Zhu et. al. (Zhu et al., 2003). They also showed interesting connections between harmonic functions and several physical and mathematical models. For example, consider a random walk that starts on an unlabeled node d on the similarity graph induced by the similarity function as described by Equation 6.12. Then $y(d)$ is equal to the probability that this random walk will hit a node labeled as 1 before it hits a node labeled as 0. In electrical engineering, $y(d)$ is equivalent to the electric potential of a node.

We can write Equation 6.12 for all documents in matrix form. Let n be the total number of documents and $\mathbf{W}$ be the (normalized) weight matrix of the similarity graph such that

$$\mathbf{W}_{ij} = \begin{cases} 0 & \text{if } i = j \\ \frac{sim(d_i, d_j)}{\sum_{d' \in N_k^{L \cup U}(d_i)} sim(d_i, d')} & \text{otherwise} \end{cases} \tag{6.13}$$

Then Equation 6.12 can be generalized to the matrix form as:

$$\mathbf{y} = \mathbf{W} \cdot \mathbf{y} \tag{6.14}$$

with the constraint that the solution vector $\mathbf{y}$ takes the value 0 or 1 for labeled documents depending on their label. Keeping this constraint in mind, we can rewrite this equation as follows:

$$(\mathbf{I} - \mathbf{W_{uu}})\mathbf{y_u} = \mathbf{W_{ul}}\mathbf{y_l} \tag{6.15}$$

where $\mathbf{y_u}$ and $\mathbf{y_l}$ denote only the values of the $\mathbf{y}$ vector that correspong to the unlabeled and the labeled documents, respectively. Similarly, $\mathbf{W_{uu}}$ is a submatrix of $\mathbf{W}$ that only includes the edges from unlabeled to unlabeled documents, and $\mathbf{W_{ul}}$ is a submatrix of $\mathbf{W}$ that only includes the edges from unlabeled to labeled documents. The only unknown in Equation 6.15 is the $\mathbf{y_u}$ vector and it can be solved efficiently with an iterative linear system solver (Zhu et al., 2003).

6.4 Experiments

In this section, we explain the experiments we performed to test the similarity functions and the learning methods introduced in the previous sections.

6.4.1 Datasets and Preprocessing

We use two standard text classification datasets. The first one is a version of the 20 Newsgroups dataset (20news-18828)[2] that contains 18,828 documents after removing the duplicates in the original dataset which has 19,997 documents. All the headers except for "From" and "Subject" are removed in the newsgroup articles. Each document belongs to exactly one newsgroup category. There are a total of 20 categories that are almost evenly distributed over the documents.

The second dataset is the Reuters-21578 dataset that consists of documents collected from the Reuters newswire in 1987. Following previous research (Joachims, 1998; Yang and Liu, 1999; Nigam et al., 2000; Rennie et al., 2003), we use the "ModApte" split of this dataset and then consider only 90 categories which have at least one training and one test document in the split. This leaves us with a total of 10789 documents (7770 training and 3019 test documents in the ModApte split). Unlike the 20 Newsgroups dataset, some of the documents in Reuters-21578 belong

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

to more than one category. The categories are also not evenly distributed: some categories have as few as two documents while others may have few thousands of documents.

We remove the stopwords and stemmed all the words in both datasets using the Porter stemmer. While computing the generation probabilities and KL divergence, the Dirichlet smoothing parameter μ is set to 1000. All this preprocessing has been performed using the Lemur toolkit (Ogilvie and Callan, 2001).

6.4.2 Implementation Details

Note that the generation probabilities and the KL divergence based similarity measure are not symmetric. As mentioned in Section 6.3.2, for a given document d, we use $p_{\text{LM}}(d|\cdot)$ and $\exp(-\text{KL}(d|\cdot))$ (not $p_{\text{LM}}(\cdot|d)$ and $\exp(-\text{KL}(\cdot|d))$) to find the nearest neighbors of d with respect to the generation probability and the KL divergence based similarity measures, respectively. However, for harmonic functions, we consider both directions and we make two documents neighbors of each other if *either* of them is in the k-nearest neighbor set of the other. The similarity value is set to the maximum of the two values for different directions. This makes the underlying graph of the harmonic function undirected.[3] In all of our nearest neighbor based experiments, k (the number of neighbors) was chosen as 30.

To find the solution of the harmonic function in Equation 6.14, we use the Iterative Template Library.[4] After computing $y(\cdot)$ in Equation 6.12 for each document, using 0.5 as the threshold value to classify a document as positive or negative does not always yield the best result. This is especially the case for datasets such as Reuters-21578 where the class sizes vary a lot. Therefore, following Zhu et. al. (Zhu

[3] Although previous research focused on undirected graphs, it can be shown that the solution of the harmonic function still exists on a directed graph. However, undirected graphs have consistently performed better in our experiments.

[4] `http://www.osl.iu.edu/research/itl/`

et al., 2003), we use *class mass normalization* to adjust the threshold for a given binary classification problem. Suppose the ratio of the training documents in the positive class to all the training documents is r. Class mass normalization classifies a document d as positive if and only if

$$z(d) = r \cdot \frac{y(d)}{\sum_{d' \in U} y(d')} - (1 - r) \cdot \frac{1 - y(d)}{\sum_{d' \in U} 1 - y(d')} \geq 0 \tag{6.16}$$

Since we have defined the semi-supervised kNN (harmonic functions) method as a binary classification algorithm, it needs to be extended to the multi-class case. For each class c in the training data, we construct a one-vs-all classifier where the training documents in c are labeled as positive and all other documents are labeled as negative. After running all these classifiers using class mass normalization, we classify each document d to the class whose classifier yields the maximum $z(d)$ value given by Equation 6.16. This extension to the multi-class case is done only for the 20 Newsgroup dataset. Since some of the documents in the Reuters corpus belong to more than one class, we construct only binary one-vs-all classifiers for each class and then evaluate them separately.

6.4.3 Results

Here we explain the metrics used to evaluate our experiments and do a critical analysis of the results.

Evaluation Metrics

The first evaluation metric we use is the classification accuracy. However, achieving high accuracy for each binary classification on the Reuters corpus is trivial since the negative class dominates the dataset. Therefore we consider precision and recall for

each binary classification defined as follows:

$$\text{Precision} = \frac{\#\text{ of correct positive predictions}}{\#\text{ of positive predictions}} \tag{6.17}$$

$$\text{Recall} = \frac{\#\text{ of correct positive predictions}}{\#\text{ of positive documents}} \tag{6.18}$$

After ranking the documents by their score in the binary classification, we have a trade-off between precision and recall by adjusting the positive/negative cutoff in the ranked list. We report the precision-recall breakeven point (Joachims, 1998) which is defined as the precision and the recall value where the two are equal. Note that the scoring transformation for harmonic functions in Equation 6.16 does not change the ranking of the documents but only the classification threshold value. Therefore, it will have an effect on the accuracy of the classification but no effect on the precision-recall breakeven point.

To combine the scores of different binary classifications for the Reuters dataset, we use microaveraging (Yang, 1999), that is, take the weighted average of the scores of all binary classifications. In other words, instead of taking the simple average of all binary classifications, the size of each class is also taken into account. This gives a better overall score since the sizes of the classes in the dataset may vary a lot.

Comparison of Nearest Neighbor Methods and Similarity Functions

Table 6.4.3 shows the accuracies of several methods on the 10 largest classes in the Reuters dataset as well as the microaverages for these 10 binary classifications and for all of the 90 binary classifications on the dataset. We see that for a given similarity measure, weighted kNN performs better than voting kNN, and harmonic functions perform better than weighted kNN overall and on most of the individual classes. KL divergence based similarity also performs better than cosine for a given learn-

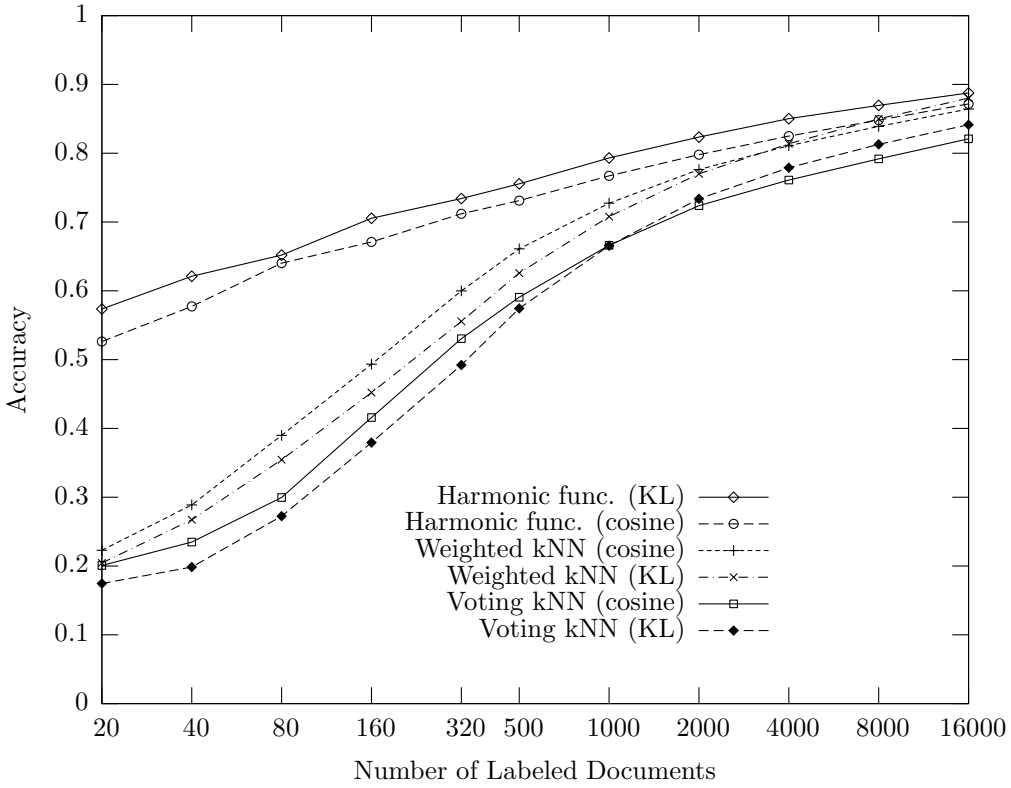


Figure 6.2: The performances of the voting kNN, weighted kNN, and semi-supervised kNN (harmonic functions) algorithms with cosine and KL similarity on the 20 Newsgroups dataset.

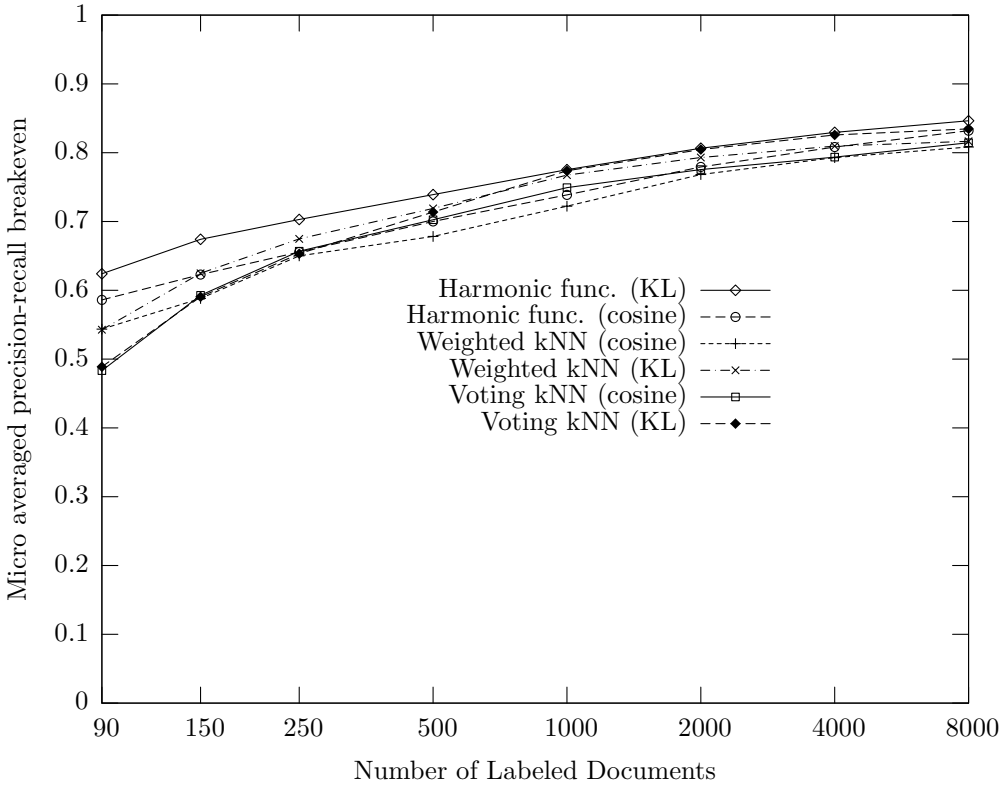


Figure 6.3: The performances of the voting kNN, weighted kNN, and semi-supervised kNN (harmonic functions) algorithms with cosine and KL similarity on the Reuters dataset.

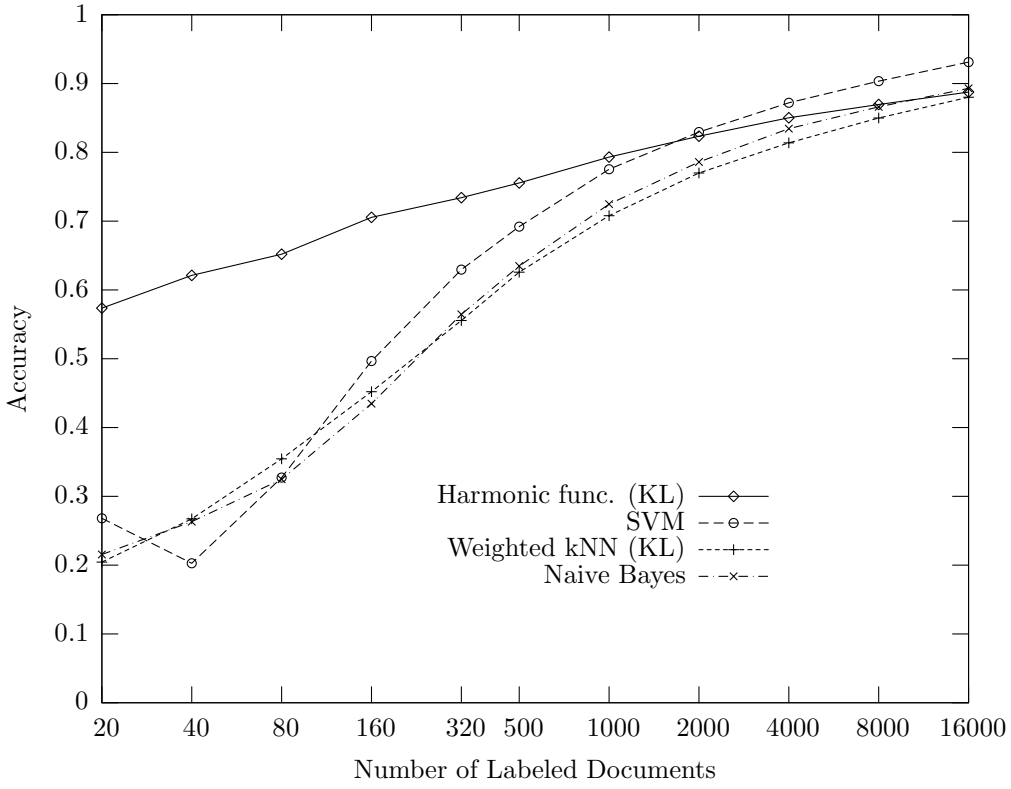


Figure 6.4: The performances of different algorithms on the 20 Newsgroups dataset with varying training data sizes.

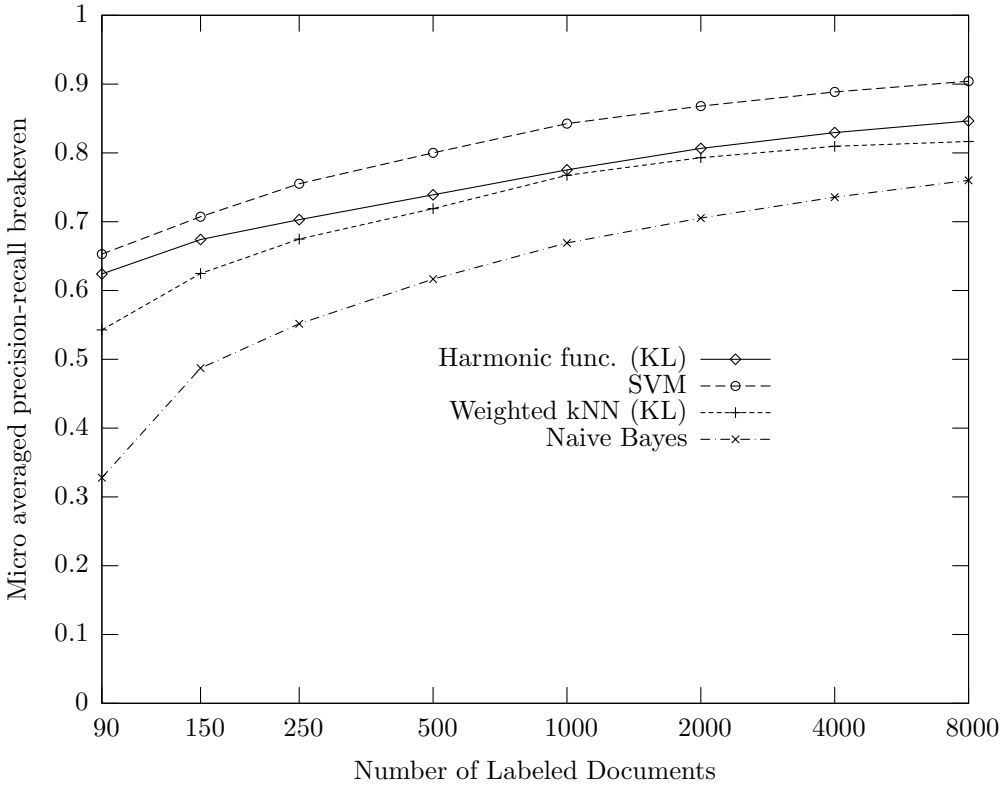


Figure 6.5: The performances of different algorithms on the Reuters dataset with varying training data sizes.

ing method. However, as we have mentioned above, using accuracy for the Reuters dataset produces very high scores because the negative class in each binary classification is too large. For this reason, we also report the precision-recall breakeven scores in Table 6.4.3. Here, we see the same pattern with the exception that weighted kNN now performs slightly (but not significantly) worse than the voting kNN.

To see the effects of semi-supervised learning, we also run several experiments by varying the size of the training data and using the rest of the dataset as the test data. For each training data size, we choose a random subset of the documents as the training set, and then we report the average of 10 such random runs. We make sure that each class is represented by at least one document in each training set. In Figure 6.2, it is clear that weighted kNN performs better than voting kNN, and harmonic functions perform better than weighted kNN on the 20 Newsgroup dataset. Not surprisingly, the differences in the accuracies of these methods tend to shrink as we introduce more training documents. One interesting observation is that KL divergence based similarity performs somewhat worse than the cosine similarity using the supervised kNN methods especially with limited training data. However, it performs the best in conjunction with harmonic functions consistently at all training data sizes. Figure 6.3 shows the results of the same experiment on the Reuters dataset using the precision-recall breakeven point microaverage of 90 classes. Similar observations can be made here except that the absolute differences among the performances are smaller. However, harmonic functions using KL divergence based similarity perform still the best especially where there is little training data.

Category	v-kNN (cos)	w-kNN (cos)	v-kNN (KL)	w-kNN (KL)	Harm. (cos)	Harm. (KL)	NB	SVM
acq	87.38	87.84	94.30	94.77	91.42	96.12	96.89	66.18
corn	98.15	98.15	98.15	98.15	98.77	98.61	95.06	95.96
crude	96.99	97.05	96.99	97.15	97.98	97.32	96.92	88.67
earn	95.26	95.26	97.88	97.88	96.42	98.48	94.37	53.53
grain	95.06	95.06	95.06	95.06	98.48	98.31	95.79	90.53
interest	96.92	97.09	96.75	96.89	97.71	97.91	94.63	92.35
money-fx	94.97	95.16	95.43	95.50	97.09	97.75	94.40	90.10
ship	97.71	98.01	98.01	98.08	98.31	98.38	97.71	95.56
trade	97.38	97.38	97.48	97.48	97.98	97.95	93.01	93.18
wheat	97.65	97.65	97.65	97.65	98.58	98.58	95.50	95.23
microavg. (10)	95.75	95.87	96.77	96.86	97.27	97.94	95.43	86.13
microavg. (90)	99.22	99.24	99.32	99.33	99.44	99.49	86.04	97.93

Table 6.2: The accuracies of various algorithms on the ModApte split of the 10 largest categories of the Reuters dataset. Averages of these categories and all of the 90 categories are also shown. v-kNN: voting kNN, w-kNN: weighted kNN, Harm.: Harmonic functions, NB: Naive Bayes, SVM: Support Vector Machines with normalized $tf \cdot idf$ document representation and linear kernel.

Comparison against Other Methods

To compare the nearest neighbor methods against other standard text classification methods, we chose two of the most popular learning algorithms: Naive Bayes and Support Vector Machines (SVM). We use the multinomial naive Bayes as explained in Section 6.2. For SVM classification, we choose the linear SVM and use the SVM^{light} package as implemented by Joachims (Joachims, 1999). We compute the tf·idf vector representation of each document normalized to unit length before running the SVM classifier.

Table 6.4.3 shows that nearest neighbor methods perform better than Naive Bayes and SVM on the Reuters dataset in terms of accuracy. Note that SVM performs much worse on some categories such as *acq* and *earn*. However, in terms of precision-recall breakeven point, SVM performs the best (Table 6.4.3).[5] This indicates that SVM's document ranking is better than other algorithms, but it may choose the

[5]The microaveraged value we have found for SVM is higher than Joachims's (Joachims, 1998) (88.62 versus 86.5) probably due to the difference in preprocessing the dataset and also the fact that we use tf·idf vectors but he uses only tf vectors.

Category	v-kNN (cos)	w-kNN (cos)	v-kNN (KL)	w-kNN (KL)	Harm. (cos)	Harm. (KL)	NB	SVM
acq	88.87	86.09	94.30	93.46	89.57	94.99	93.05	97.50
corn	71.43	67.86	62.50	62.50	67.86	66.07	46.43	87.50
crude	83.07	83.60	85.71	85.19	87.30	88.36	84.13	89.42
earn	93.65	95.31	97.52	97.33	95.03	97.88	94.85	98.80
grain	83.22	83.22	81.21	80.54	85.23	83.22	73.15	92.62
interest	71.76	73.28	76.34	75.57	77.10	79.39	61.07	81.68
money-fx	70.39	69.27	79.33	77.65	77.09	78.77	65.92	79.89
ship	79.78	78.65	80.90	79.78	78.65	80.90	79.78	88.76
trade	74.36	71.79	74.36	75.21	77.78	74.36	57.27	76.92
wheat	67.61	67.61	64.79	64.79	74.65	69.01	63.38	85.92
microavg. (10)	86.26	86.01	89.81	89.31	88.27	90.71	85.22	93.68
microavg. (90)	80.07	79.89	82.61	82.18	82.05	83.89	74.63	88.62

Table 6.3: The precision-recall breakeven points of various algorithms on the ModApte split of the 10 largest categories of the Reuters dataset. Microaverages of these categories and all of the 90 categories are also shown. v-kNN: voting kNN, w-kNN: weighted kNN, Harm.: Harmonic functions, NB: Naive Bayes, SVM: Support Vector Machines with normalized $tf \cdot idf$ document representation.

wrong decision boundary which results in low accuracy. Once again, Naive Bayes performs worse than all of the nearest neighbor algorithms.

Figures 6.4 and 6.5 show the comparison of these algorithms at varying training data sizes on the 20 Newsgroups and Reuters datasets, respectively. For simplicity, we only include harmonic functions and weighted kNN with KL divergence based similarity from Figures 6.2 and 6.3. Semi-supervised harmonic functions perform much better on the 20 Newsgroups dataset than all other algorithms at smaller training data sizes. SVM seems to catch up only when a lot more training documents are introduced. Weighted kNN's performance is competitive and parallel to Naive Bayes's. On the Reuters dataset, however, SVM is the best algorithm at all training sizes. Supervised and semi-supervised nearest neighbor methods are still better than Naive Bayes this time by a wide margin.

6.5 Conclusion

We have presented an extensive evaluation of several nearest neighbor methods for text classification. Despite their simplicity, nearest neighbor based approaches perform quite well on text classification. We have shown that this performance can be improved even more by making use of new similarity metrics motivated by the language modeling approach in information retrieval. The results are often better than Naive Bayes and very competitive to the state-of-the-art SVM text classification.

The nearest neighbor framework can be extended to the semi-supervised case. We have shown that this results in an algorithm that is equivalent to the semi-supervised learning harmonic functions and presented the first extensive evaluation of this algorithm on text categorization. The results, especially with limited training data, are remarkable surpassing SVM on one of the datasets we have tried. As a future work, we want to compare this semi-supervised method to the semi-supervised counterparts of other algorithms such as Naive Bayes extended with EM and transductive SVM.

CHAPTER VII

CONCLUSION

We have presented a new graph-based framework for text collections. The fundamental data structure in this framework is a *text similarity graph* where nodes are textual units such as sentences or documents, and the edges between the nodes are induced by a text similarity function. Graphs provide a global view of a text collection so that we can make global inferences about the collection while performing local computations on the relationships among the nodes. For example, with a local definition of the importance of a node stated only in terms of its neighbors in a graph, we can compute a global ranking since local computations recursively affect all nodes connected in the graph. Such a ranking method, *LexRank*, performed well in Chapter III to rank sentences for the extractive text summarization problem. Our graph-based summarization method ranked as one of the top performing systems in the Document Understanding Conferences evaluations. It has also inspired other graph-based summarization systems since its introduction.

A related benefit of graph-based computation is that we can make inferences about nodes that are not directly connected in the graph by one edge. For our text similarity graphs, this means that we can induce similarity values for textual units that are not obviously similar with respect to a simple similarity function such as *cosine*. In

Chapter IV, we have introduced a modified version of LexRank to rank the sentences in a collection based on a given query. By using random walks on the graph, we can propagate the similarity information from the query to the sentences so that we can both produce a ranking similar to LexRank and bias this ranking towards the query at the same time. The effect is similar to that of *relevance feedback* in information retrieval, but our method provides a unified framework of retrieval and feedback information, and is more principled and easier to tune. The method is also very general and can be used for several retrieval problems such as focused summarization (or passage retrieval) in Chapter IV, question answering or classical document retrieval.

In Chapter V, we tested the new similarity values induced by random walks in the document clustering problem. We achieved statistically significant improvements in clustering accuracy over using the base similarities without the random walks. Furthermore, the improvements got monotonically better as we looked at longer random walks except that paths longer than three edges gave us no additional improvement. This gave us important clues about the limit of the transitivity of similarity in text and how far we can propagate it in a graph to get better approximations to real semantic similarities between the nodes.

We revisited the nearest neighbor-based approaches to text classification in Chapter VI. With the intuitions we used in previous chapters, we derived a semi-supervised version of the k-nearest neighbor algorithm based on random walks that is equivalent to the harmonic functions algorithm previously introduced elsewhere. We achieved significant improvements over the classical versions of the k-nearest neighbor algorithm, rivaling state-of-the-art classification methods such as Support Vector Machines.

Another contribution in this thesis has been the use of language modeling-based similarity functions for the first time in the text summarization, document clustering and text classification problems. Language modeling approach performed consistently better than the classical cosine measure used in these problems achieving improvements as high as 25% in document clustering accuracy. Our graph-based approach using random walks achieved additional significant improvement on top of this.

It is important to point out that all of our methods introduced in this thesis are language-independent and knowledge-lean methods that are easy to apply in a diverse set of problem settings. They do not depend on any language specific information or external knowledge sources other than the input text. However, such methods based on external knowledge are orthogonal to our study and may further improve our results reported here.

BIBLIOGRAPHY

BIBLIOGRAPHY

James Allan, Courtney Wade, and Alvaro Bolivar. 2003. Retrieval and novelty detection at the sentence level. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 314–321, Toronto, Canada. ACM Press.

Regina Barzilay and Michael Elhadad. 1997. Using Lexical Chains for Text Summarization. In *Proceedings of the ACL/EACL'97 Workshop on Intelligent Scalable Text Summarization*, pages 10–17, Madrid, Spain, July.

Regina Barzilay and Michael Elhadad. 1999. Using Lexical Chains for Text Summarization. In Inderjeet Mani and Mark T. Maybury, editors, *Advances in Automatic Text Summarization*, pages 111–121. The MIT Press.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of HLT-NAACL*.

P.B. Baxendale. 1958. Man-made index for technical litterature - an experiment. *IBM J. Res. Dev.*, 2(4):354–361.

Ron Bekkerman, Andrew McCallum, and Gary Huang. 2004. Automatic categorization of email into folders: Benchmark experiments on Enron and SRI corpora. Technical Report IR-418, Center of Intelligent Information Retrieval, UMass Amherst.

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

Ron Brandow, Karl Mitze, and Lisa F. Rau. 1995. Automatic condensation of electronic publications by sentence selection. *Information Processing and Management*, 31(5):675–685.

Chris Brew and Sabine Schulte im Walde. 2002. Spectral clustering for german verbs. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117.

Jaime G. Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Research and Development in Information Retrieval*, pages 335–336.

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.

Kenneth W. Church and William Gale. 1991. A comparison of the enhanced good-turing and deleted estimation methods for estimating probabilities of english bigrams. *Computer Speech and Language*, 5(1):19–54, January.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*.

Thomas M. Cover and Joy A. Thomas. 1991. *Elements of Information Theory*. Wiley.

Hal Daumé III and Daniel Marcu. 2004. A phrase-based hmm approach to document/abstract alignment. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 119–126, Barcelona, Spain, July. Association for Computational Linguistics.

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.

Thomas G. Dietterich. 2000. Ensemble methods in machine learning. In Josef Kittler and Fabio Roli, editors, *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer.

Sergei N. Dorogovtsev and Jose F. F. Mendes. 2001. Language as an evolving word web. *Proceedings of the Royal Society of London B*, 268(1485):2603–2606, December 22.

Peter G. Doyle and J. Laurie Snell. 1984. *Random Walks and Electric Networks*. Mathematical Association of America.

H.P. Edmundson. 1969. New Methods in Automatic Extracting. *Journal of the Association for Computing Machinery*, 16(2):264–285, April.

Güneş Erkan and Dragomir R. Radev. 2004a. Lexpagerank: Prestige in multi-document text summarization. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 365–371, Barcelona, Spain, July. Association for Computational Linguistics.

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

Güneş Erkan and Dragomir R. Radev. 2004c. The University of Michigan at DUC 2004. In *Proceedings of the Document Understanding Conferences*, Boston, MA, May.

Güneş Erkan. 2006a. Language model-based document clustering using random walks. In Robert C. Moore, Jeff A. Bilmes, Jennifer Chu-Carroll, and Mark Sanderson, editors, *HLT-NAACL*. The Association for Computational Linguistics.

Güneş Erkan. 2006b. Using biased random walks for focused summarization. In *Proceedings of Document Understanding Conference (DUC)*, New York City, NY.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press.

Ramon Ferrer i Cancho and Ricard V. Solé. 2001. The small world of human language. *Proceedings of the Royal Society of London B*, 268(1482):2261–2265, November 7.

Robert Gaizauskas, Mark Hepple, and Mark Greenwood. 2004. Information Retrieval for Question Answering: a SIGIR 2004 Workshop. In *SIGIR 2004 Workshop on Information Retrieval for Question Answering*.

I. J. Good. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40:237–264.

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.

Vasileios Hatzivassiloglou, Judith L. Klavans, Melissa L. Holcombe, Regina Barzilay, Min yen Kan, and Kathleen R. Mckeown. 2001. Simfinder: A flexible clustering tool for summarization.

Bruno Jedynak and Damianos Karakos. 2007. Unigram language models using diffusion smoothing over graphs. In *Proceedings of the Second Workshop on TextGraphs: Graph-Based Algorithms for Natural Language Processing*, pages 33–36, Rochester, NY, USA. Association for Computational Linguistics.

Harold Jeffreys. 1961. *Theory of Probability*. Oxford: Carendon Press.

Frederick Jelinek and Robert L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. *Pattern Recognition in Practice*, pages 381–397.

Hongyan Jing. 2002. Using hidden markov modeling to decompose Human-Written summaries. *CL*, 28(4):527–543.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*. Springer.

Thorsten Joachims. 1999. Making large–scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA. MIT Press.

W. E. Johnson. 1932. Probability: The deductive and inductive problems. *Mind*, 41(164):409–423.

Slava M. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401, March.

Kenrick Mock. 2001. An experimental framework for email categorization and management. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 392–393.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization — step one: Sentence compression. In *Proceeding of The 17th National Conference of the American Association for Artificial Intelligence (AAAI-2000)*, pages 703–710.

Ron Kohavi, Pat Langley, and Yeogirl Yun. 1997. The utility of feature weighting in nearest-neighbor algorithms. In *Proceedings of the Ninth European Conference on Machine Learning (ECML-97)*, Prague. Springer Verlag.

Julian Kupiec, Jan O. Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Research and Development in Information Retrieval*, pages 68–73.

Oren Kurland and Lillian Lee. 2005. Pagerank without hyperlinks: structural re-ranking using links induced by language models. In Ricardo A. Baeza-Yates, Nivio Ziviani, Gary Marchionini, Alistair Moffat, and John Tait, editors, *SIGIR*, pages 306–313. ACM.

John Lafferty and Chengxiang Zhai. 2001. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 111–119.

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.

Lillian Lee. 1997. *Similarity-Based Approaches to Natural Language Processing*. Ph.D. thesis, Harvard University, Cambridge, MA.

Daniel S. Leite, Lucia H. M. Rino, Thiago A. S. Pardo, and Maria das Graças V. Nunes. 2007. Extractive automatic summarization: Does more linguistic knowledge make a difference? In *Proceedings of the Second Workshop on TextGraphs: Graph-Based Algorithms for Natural Language Processing*, pages 17–24, Rochester, NY, USA. Association for Computational Linguistics.

G. Lidstone. 1920. Note on the general case of the bayes-laplace formula for inductive or a posteriori probabilities. *Transactions of the Faculty of Actuaries*, pages 8:182–192.

Chin-Yew Lin and E.H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence. In *Proceedings of 2003 Language Technology Conference (HLT-NAACL 2003), Edmonton, Canada, May 27 - June 1*.

Ziheng Lin and Min-Yen Kan. 2007. Timestamped graphs: Evolutionary models of text for multi-document summarization. In *Proceedings of the Second Workshop on TextGraphs: Graph-Based Algorithms for Natural Language Processing*, pages 25–32, Rochester, NY, USA. Association for Computational Linguistics.

Jianhua Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145.

Chin-Yew Lin. 1999. Training a Selection Function for Extraction. In *Proceedings of the Eighteenth Annual International ACM Conference on Information and Knowledge Management (CIKM)*, pages 55–62, Kansas City, November -6. ACM.

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

H.P. Luhn. 1958. The Automatic Creation of Literature Abstracts. *IBM Journal of Research Development*, 2(2):159–165.

David J. C. MacKay and Linda C. Bauman Peto. 1995. A hierarchical Dirichlet language model. *Natural Language Engineering*, 1(3):1–19.

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.

Inderjeet Mani and Eric Bloedorn. 1997. Multi-document summarization by graph search and matching. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 622–628, Providence, Rhode Island. American Association for Artificial Intelligence.

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

Kathleen R. McKeown, Regina Barzilay, David Evans, Vasileios Hatzivassiloglou, Simone Teufel, Yen M. Kan, and Barry Schiffman. 2001. Columbia Multi-Document Summarization: Approach and Evaluation. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New Orleans, LA.

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, Paul Tarau, and Elizabeth Figa. 2004. Pagerank on semantic networks, with application to word sense disambiguation. In *Proceedings of The 20st International Conference on Computational Linguistics (COLING 2004)*, Geneva, Switzerland, August.

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.

Marie-Francine Moens, Caroline Uyttendaele, and Jos Dumortier. 1999. Abstracting of legal cases: the potential of clustering based on the selection of representative objects. *J. Am. Soc. Inf. Sci.*, 50(2):151–161.

Jane Morris and Graeme Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17(1):21–48.

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.

Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103.

Paul Ogilvie and James P. Callan. 2001. Experiments using the lemur toolkit. In *TREC*.

David Opitz and Richard Maclin. 1999. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198.

Miles Osborne. 2002. Using Maximum Entropy for Sentence Extraction. In *ACL Workshop on Text Summarization*, July 12–13,.

Jahna Otterbacher, Güneş Erkan, and Dragomir Radev. 2005. Using random walks for question-focused sentence retrieval. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 915–922, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The pagerank citation ranking: Bringing order to the web. *Technical report, Stanford University, Stanford, CA*.

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

Dragomir R. Radev and Kathleen R. McKeown. 1998. Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, 24(3):469–500, September.

Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *ANLP/NAACL Workshop on Summarization*, Seattle, WA, April.

Dragomir Radev, Sasha Blair-Goldensohn, and Zhu Zhang. 2001. Experiments in single and multi-document summarization using MEAD. In *First Document Understanding Conference*, New Orleans, LA, September.

Dragomir Radev. 2000. A common theory of information fusion from multiple text sources, step one: Cross-document structure. In *Proceedings, 1st ACL SIGDIAL Workshop on Discourse and Dialogue*, Hong Kong, October.

Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David Karger. 2003. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of ICML-03, 20th International Conference on Machine Learning*, Washington, DC. Morgan Kaufmann Publishers, San Francisco, US.

Jason D. M. Rennie. 2001. Improving multi-class text classification with naive bayes. In *MIT AI-TR*.

Gerard Salton and Chris Buckley. 1990. Improving retrieval performance by relevance feedback. *JASIS*, 41(4):288–297.

Gerard Salton and Michael J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw Hill.

Gerard Salton, Amit Singhal, Mandar Mitra, and Chris Buckley. 1997. Automatic Text Structuring and Summarization. *Information Processing & Management*, 33(2):193–207.

Hinrich Schütze, David A. Hull, and Jan O. Pedersen. 1995. A comparison of classifiers and document representations for the routing problem. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval*, pages 229–237, Seattle, US. ACM Press, New York, US.

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

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

Karen year = Sparck-Jones. A statistical interpretation of term specificity and its application in retrieval.

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, Chris Manning, and Andrew Ng. 2004. Learning random walk models for inducing word dependency distributions. In *Proceedings of ICML*.

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.

Michael Witbrock and Vibhu O. Mittal. 1999. Ultra-Summarization: A Statistical Approach to Generating Highly Condensed Non-Extractive Summaries. In *SIGIR99*, pages 315–316, Berkeley, CA.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Speech IR & Text Categorization, pages 42–49.

Yiming Yang. 1999. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1-2):69–90.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*.

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. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 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.

Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In Tom Fawcett and Nina Mishra, editors, *ICML*, pages 912–919. AAAI Press.