Unsupervised Graph-Based Similarity Learning Using Heterogeneous Features

by

Pradeep Muthukrishnan

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
2011

Dedicated to my parents

ACKNOWLEDGEMENTS

First and foremost, I am truly grateful to my advisor Dragomir Radev. This thesis would not have been possible without him and without the support he has given me over the past few years. His constant guidance, motivation and patience has been essential in my development as a scientist. Also, I would like to thank Qiaozhu Mei for many insightful discussions and feedback that helped shape and improve this work. I would also like to thank the other members of my dissertation committee, Steve Abney, Honglak Lee and Zeeshan Syed for their careful criticism and insightful feedback that improved the quality of this work.

I would like to thank all members of the Computational Linguistics and Information Retrieval (CLAIR) group in the University of Michigan, Vahed Qazvinian, Ahmed Hassan, Arzucan Ozgur, and Amjad Abu-Jbara for being great colleagues and many insightful discussions that we had together. Special thanks to my roommates Kumar Sricharan and Gowtham Bellala for the innumerable late-night discussions on my research.

My heartfelt thanks to all my friends who have been family for me here in Ann Arbor, especially Avani Kamath, Tejas Jayaraman and Gandharv Kashinath. I would also like to thank all my friends in India, especially Jason Lazard, Zeeshan Qureshi and Roshan Muthaiya.

Words cannot express my gratitude to my parents. Not a single page of this thesis would have been possible without their great sacrifices and constant support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

ABSTRACT

Unsupervised Graph-Based Similarity Learning Using Heterogeneous Features

by

Pradeep Muthukrishnan

Chair: Dragomir Radkov Radev

Relational data refers to data that contains explicit relations among objects. Nowadays, relational data are universal and have a broad appeal in many different application domains. The problem of estimating similarity between objects is a core requirement for many standard Machine Learning (ML), Natural Language Processing (NLP) and Information Retrieval (IR) problems such as clustering, classification, word sense disambiguation, etc. Traditional machine learning approaches represent the data using simple, concise representations such as feature vectors. While this works very well for homogeneous data, i.e, data with a single feature type such as text, it does not exploit the availability of different feature types *fully*. For example, scientific publications have text, citations, authorship information, venue information. Each of the features can be used for estimating similarity. Representing such objects has been a key issue in efficient mining (Getoor and Taskar, 2007). In this thesis, we propose natural representations for relational data using multiple, connected layers of graphs; one for each feature type. Also, we propose novel algorithms for estimating similarity using multiple heterogeneous features. Also, we present novel algorithms for tasks like topic detection and music recommendation using the estimated similarity measure. We demonstrate superior performance of the proposed algorithms (root mean squared error of 24.81 on the Yahoo! KDD Music recommendation data set and classification accuracy of 88% on the ACL Anthology Network data set) over many of the state of the art algorithms, such as Latent Semantic Analysis (LSA), Multiple Kernel Learning (MKL) and spectral clustering and baselines on large, standard data sets.

CHAPTER I

Introduction

1.1 Introduction

Similarity is a measure of symmetry or resemblance between two concepts or objects. It is a fundamental concept that exists since Aristotle. Aristotle stated *the study of what is similar is useful for inductive reasoning because it is by induction of particulars on the basis of similars that we claim to bring in the universal.*

Most machine learning algorithms represent the data using simple forms like feature vectors. For example, a document D in a corpus of M documents can be represented using a $\vec{V}(D) \in \mathbb{R}^N$ where N is the total number of unique words in the corpus. $\vec{V}(D)_i$ represents the presence of the i^{th} term in document D.

While such a simple, concise representation form helps reduce space usage, they are not very amenable for making complex inferences and learning, in general. For example, publications, in addition to text, has many relational features like citations, authorship information, venue information, etc. Another example is videos found on the Internet. Usually, they have audio, video and a small text description, additional metadata information like the date of upload, uploader information, etc. Also, feature vectors cannot be used for representing linked relational data. For example, citations between two publications or links between two videos, etc.

Similarity plays a very important part in many Natural Language Processing (NLP) and Information Retrieval (IR) tasks. It has been used for tasks like classification, clustering, etc. Since similarity is such a fundamental concept, not surprisingly, there are many different algorithms to compute similarity between different objects described by different features. However, it is hard to evaluate similarity measures in isolation. In fact, it is hard to even rigorously define similarity from first principles. Thus, most of the algorithms to compute similarity scores are evaluated extrinsically, i.e, the similarity scores are used for an external task like clustering or classification and the performance on the external task is used as the performance measure for the similarity measure. This approach of evaluation also shows the different applications of the learned similarity measure.

Clustering is the task of grouping data points into clusters such that data points within the same cluster are similar to each other while data points across two different clusters are dissimilar. Classification is the task of labeling all data points with a label representing the category it belongs to. The key difference between clustering and classification is the availability of training data. Classification is almost always a supervised task while clustering is an unsupervised task. In graph clustering/classification,the input is a graph, $G = (V, E)$ where V is the set of data points, represented as nodes of the graph and the edge weight between two nodes, $w(u, v) \rightarrow \mathbb{R}^+ : (u, v) \in E$, represent the similarity between the two nodes. In general, the computed similarity values are normalized so that all similarity values are $0 \leq w(u, v) \leq 1$. If the computed similarity values are ideal, i.e, the similarity between two data points belonging to the same cluster is 1 while the similarity between two data points across two different clusters is 0, then the different connected components (after removing zero-weighted edges) are the different clusters. Thus, computing the similarity between the data points is an integral subtask for clustering which makes clustering an ideal task for evaluating similarity measures. In this thesis, we study different existing similarity measures in the context of clustering and classification.

In this thesis, we focus on learning similarity measures using multiple heterogeneous features like in the examples mentioned above. The problem setup is very general: An object O is represented using m different (possibly) heterogeneous feature types , $F = \{F_1, F_2, \ldots, F_m\}$. Each feature type, F_i consists of a set of features, $F_i = \{f_{1i}, f_{2i}, \ldots, f_{n_i i}\}$. Also, we assume the existence of m initial similarity measures, S_i ($\forall i = 1, 2, \ldots, m$. S_i) assigns a similarity value for any two features of feature type i: $S(f_{ji}, f_{ki}) \rightarrow \mathbb{R}$. The task is to refine all the similarity measures using the available relational data.

The rest of the thesis is organized as follows. In Chapter II, we start with a short survey of different string similarity measures, graph similarity measures and the clustering and classification algorithms that will be used later for evaluating the similarity measures

In Chapter III, we look at methods to augment existing similarity measures using the different relations. The motivation behind this work is that, in many domains, individual similarity measures cannot completely capture the true content similarity.

However, there are many independent sources of similarity because of the availability of relational data. We can exploit the independence between the similarity measures to construct a superior similarity measure. The ACL Anthology Network (AAN) (Radev et al., 2009b) is a manually curated networked database of citations, collaborations, and summaries in the field of Computational Linguistics. The full AAN includes the raw text of all the papers in addition to full citation and collaboration networks. (See Appendix B for a full description of the AAN data set). Consider the following three publications from the AAN data set

(1) Peter F. Brown, Vincent J. Della Pietra., Stephen A. Della Pietra, Robert L. Mercer. The Mathematics Of Statistical Machine Translation: Parameter Estimation. Computational Linguistics. 1993.

(2) Franz Josef Och. Minimum Error Rate Training In Statistical Machine Translation. In Proceedings of ACL. 2003.

(3) William A. Gale, Kenneth Ward Church. A Program For Aligning Sentences In Bilingual Corpora. In Proceedings of ACL. 1991.

(4) Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Meredith J. Goldsmith, Jan Hajic, Robert L. Mercer, Surya Mohanty. But Dictionaries Are Data Too. In Proceedings of HLT. 1993.

Publications (1) and (2) can be deemed similar using simple string similarity measures. However, publications (3) and (4) also belong to the field of machine translation but mention the phrase *machine translation* only once in their entire text.

So to attribute a high similarity value between the different pairs of publications we need to look at similarity due to other relations. For example, the outgoing citation from publication (3) to publication (1) can be exploited to learn a non-zero similarity score between the two publications. Publication (1) and (4) have a lot of common authors and hence using the hypothesis that authors write similar papers, we can infer the similarity between them. Also, once we find that many papers written by two authors are similar, we can increment the similarity value between the two authors. This can further lead to inferring similarity between some other papers written by them. Thus, we can learn how to compute citation similarity from author similarity, keyword similarity and other similarity measures and vice-versa. We refer to this learning as learning across feature types. We can also learn from the transitivity of similarity measures which is best explained by the following example. For example, consider the following three sentences

(5) Machine Translation is an active research field.

(6) Machine Translation makes use of bilingual grammars.

(7) We use the English-Spanish bilingual grammar.

Most string similarity measures depend on word overlap. Therefore, sentences (5) and (6) share some similarity, and sentences (6) and (7) share some similarity while (5) and (7) do not share any similarity. But, clearly, (5) and (7) share some semantic similarity and this can be inferred through the transitivity of the similarity relation. This is achieved by using graph-based methods similar to label propagation and graph regularization. Although the example is based on textual similarity, sparsity of similarity relations in any domain can be dealt with using the proposed method.

In Chapter IV, we extend the proposed framework to allow simultaneous optimization of feature weights and similarity learning. Estimation of similarity between objects directly relies on the feature weights. For example, once we learn publications 4) and 5) also belong to the field of machine translation, we can use this information to increase the feature weight of the keyword "machine translation" for publications 4) and 5). We also provide elaborate evaluations on three different tasks, clustering, classification and music recommendation. We compare the proposed similarity measure with different baselines, state-of-the-art similarity measures based on multiple kernel learning (Bach et al., 2004), Latent Semantic Analysis (LSA) (Deerwester et al., 1990).

In Chapter V, we address a different problem, facet detection in blogs using the proposed framework. A high-profile news event is usually followed by an outburst of blog posts which can be mined for information related to the event. Specifically, we identify a diverse, representative set of facets of any blog story. Each facet is represented by a keyphrase. For example, let the news story be the *Tragic Virginia Tech Shooting*, then the keyphrase "'Cho-Seung Hui"' is a facet/topic. We formulate the problem of choosing the set of phrases as an optimization problem over two feature types, the set of documents and the keyphrases themselves. We also have similarity estimates between the different feature types, keyphrases and documents. There exists an edge between a keyphrase and a document if the document contains the keyphrase. The task is to choose a set of facets such that all the important topics of the news event are represented while no two chosen facets are too similar to each other. We develop adaptive algorithms for this task in the proposed framework.

The key contributions of the thesis are as follows.

1. A representation model for representing objects with heterogeneous feature types for efficiently estimating similarity. The proposed model has the following advantages over feature vectors:

   - The model is generic and is capable of representing different types of features, including nominal, discrete, real-valued and link-based features.
   - The model is capable of representing a wide variety of dependencies between different features. For example, if there is information available regarding different feature types' contribution to the overall similarity between objects, it can be easily incorporated.
   - The model allows learning across feature types. For example, it can be used to learn similarity between publications using similarity measures between authors, keywords and venues and vice-versa.

2. A regularization framework for unifying different similarity measures and learning feature weights.

3. Completely unsupervised algorithms in the proposed framework to efficiently estimate feature weights and compute similarity between objects with many heterogeneous feature types.

4. Novel algorithms for tasks like music recommendation and topic detection using the proposed similarity measures.

In Appendix A, we propose an active classification algorithm which augments existing similarity measures. Existing similarity measures like cosine similarity does not exploit higher order dependencies. Consider the problem of document classification. It is possible for two documents to be semantically similar but because of vocabulary differences the two documents are assigned a very low similarity score. We can infer that two documents are similar if there are many other documents that are similar to both documents. We represent objects to be classified using a graph, where the set of nodes correspond to the set of objects and the edge weights represent similarity between the objects. We increment the similarity value between two nodes in the graph if many high similarity paths exist between them. We also present active learning algorithms to pick representative nodes in a graph to be labeled by the user. We compare the proposed active learning algorithm to existing active graph clustering algorithms on many standard benchmarked data sets.

In Appendix B, we describe the ACL Anthology Network (AAN), a networked database of publications. Citation data was obtained using text extraction from a collection of PDF les with significant manual post-processing performed to clean up the results. Manual annotation of the references was then performed to complete the citation network. We also describe the derived data set from AAN which can be used for further research in relational data learning.

CHAPTER II

Related Work

2.1 Introduction

Similarity can be defined as "some degree of symmetry in either analogy or resemblance between two or more concepts or objects". A similarity measure quantifies this notion of similarity, typically as a scalar value. Estimating similarity is one of the oldest problems in Computer Science (Tversky, 1977; Findler and van Leeuwen, 1979; Nakatsu et al., 1982). The importance of similarity measures is rather obvious given the number of problems that use similarity measures as a tool.

Similarity has different meanings in different fields. In Geometry, two objects are considered to be similar if the objects are identical except for uniform scaling. In the field of Philosophy, two objects are similar if they possess similar characteristics/traits. For example, the sports Tennis and Squash can be considered similar because a racquet is used in both the sports. In the field of publications, two publications can be considered similar if they belong to the same field of research. For example, let p_1, p_2 be two publications on statistical machine translation, p_3 be a publication on machine translation and p_4 be a publication on convex optimization. Let Similarity between two publications, p_i and p_j, be represented as $S(p_i, p_j)$. Then the similarity measure should be such that $S(p_1, p_2) > S(p_1, p_3)$ and $S(p_1, p_3) > S(p_1, p_4)$.

In general, computing a similarity score requires identifying exact or approximate matches of patterns in the objects being compared. For example, in the case of documents, we look for matches of terms/words. Later, we will see in detail how the different notions of similarity will be exploited to compute similarity scores between documents, words and structured objects. The rest of the chapter is organized as follows. The next four sections will cover the different similarity measures for simple objects, words, documents and structured objects. Later, we go over the results of evaluation of the different similarity measures.

2.2 Vector Similarity

Let $\vec{V_i}$ and $\vec{V_j}$ be two N-dimensional vectors, that is, $\vec{V_i}, \vec{V_j} \in \mathbb{R}^N$. The similarity between the two vectors can be computed using the following methods.

1. Inner Product: The formula for the inner product similarity between the two vectors, $\vec{V_i}$ and $\vec{V_j}$ is

$$Sim(\vec{V_i}, \vec{V_j}) = \sum_{k=1}^{N} V_{ik} \cdot V_{jk} \tag{2.1}$$

The problem with this formulation is that the similarity score is not upper bounded and hence it is difficult to interpret the similarity value. To avoid this problem, a simple solution is to normalize the document vector using the L2 norm. The resulting measure is called as the cosine similarity.

2. Cosine Similarity (Salton et al., 1975):

$$Sim_{ij} = \frac{\sum_{k=1}^{N} V_{ik} \times V_{jk}}{\sqrt{\sum_{k=1}^{N} {V_{ik}}^2} \times \sqrt{\sum_{k=1}^{N} {V_{jk}}^2}} \tag{2.2}$$

Notice that this is in line with the notion of similarity in the field of Geometry, with a pair of document vectors having a similarity score of 1 if one of the vectors is a uniformly scaled version of the other. The similarity between two vectors is equal to the cosine of the angle between the two vectors.

2.3 Set Similarity

A set is a collection of distinct objects. Let X and Y be two sets containing n, m elements, respectively. Then similarity between the two sets can be computed using the following methods

1. Jaccard Index (Jaccard, 1901): The Jaccard similarity is defined as:

$$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}. \tag{2.3}$$

2. Dice Coefficient (Dice, 1945): The Dice coefficient, named after Lee Raymond Dice, is defined as,

$$S = \frac{2 \cdot |X \cap Y|}{|X| + |Y|}. \tag{2.4}$$

Note that the Dice coefficient is related to the Jaccard Index as, $S = \frac{2 \cdot J}{1+J}$.

2.4 Word Similarity

2.4.1 String Match-based Similarity

In the free text representation, similarity is proportional to the number of common characters between the two strings. Some of the earliest string similarity measures were based on substring matches Longest Common Subsequence (Nakatsu et al., 1982), (Kashyap and Oommen, 1983), Edit Distance or Levenshtein distance (Levenshtein, 1966a).

Let two strings be defined as follows: $X = (x_1, x_2 \ldots, x_m)$ and $Y = (y_1, y_2 \ldots y_n)$. Let the length of the longest common substring be k_1, then the susbtring similarity between X and Y is defined as

$$S_{substring}(X, Y) = \frac{k_1}{max(N, M)} \tag{2.5}$$

For example, the similarity between the two words "micro-computer" and "computerized" which have the common substring "computer", is $\frac{8}{14}$.

A subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements. For example, ABD is a subsequence of ABCDEF. However, note that substring and subsequence are not synonyms. Substrings are consecutive parts of a string, while subsequences need not be. Subsequence based matching is used in bioinformatics, for computing similarity between gene sequences. The similarity is proportional to the longest common subsequence between X and Y. Let the length of the longest common subsequence be k_2. The subsequence similarity is defined as

$$S_{\text{subsequence}}(X, Y) = \frac{k_2}{max(N, M)} \tag{2.6}$$

For example, the subsequence similarity between the two DNA sequences, $S_1 =$ AGCAT and $S_2 =$ GAC is $\frac{2}{5}$. A naïve computation of the longest common sequence between two sequences takes exponential time, $O(2^{max(N,M)})$. However, it can be more efficiently computed using dynamic programming in quadratic time $O(N * M)$.

2.4.2 Knowledge-Based Approaches

WordNet(Miller, 1995) is a lexical database for the English language. It groups English words into sets of synonyms called *synsets*, provides short, general definitions, and records the various semantic relations between these synonym sets. The recorded semantic relations vary based on the type of word. Some of the commonly used relations in automatic text analysis are,

- Nouns
  - hypernyms: Y is a hypernym of X if every X is a (kind of) Y (canine is a hypernym of dog, because every dog is a member of the larger category of canines)
  - hyponyms: Y is a hyponym of X if every Y is a (kind of) X (dog is a hyponym of canine)
  - coordinate terms: Y is a coordinate term of X if X and Y share a hypernym (wolf is a coordinate term of dog, and dog is a coordinate term of

wolf)

  - holonym: Y is a holonym of X if X is a part of Y (building is a holonym of window)
  - meronym: Y is a meronym of X if Y is a part of X (window is a meronym of building)

- Verbs

  - hypernym: The verb Y is a hypernym of the verb X if the activity X is a (kind of) Y (to perceive is an hypernym of to listen)
  - troponym: The verb Y is a troponym of the verb X if the activity Y is doing X in some manner (to lisp is a troponym of to talk)
  - entailment: The verb Y is entailed by X if by doing X you must be doing Y (to sleep is entailed by to snore)
  - coordinate terms: Those verbs sharing a common hypernym (to lisp and to yell)

Knowledge-based approaches quantify the degree to which two words are semantically related using information drawn from semantic networks like WordNet. All the metrics we will discuss compute similarity between "Concepts" or "Synsets". A Synset or a Concept is represented by a set of words, each of which has a sense that names that concept (and each of which is therefore synonymous with the other words in the Synset).

Leacock-Chodorow : (Leacock and Chodorow, 1998) relies on the shortest path between two synsets in WordNet's taxonomy for their measure of similarity. However, they limit their attention to hypernymy links (other links are discarded when finding the path between the two synsets) and scale the path length by the overall depth $D = 16$ of the WordNet taxonomy:

$$sim_{LC}(c_1, c_2) = -\log \frac{path\_length(c1; c2)}{2D} \tag{2.7}$$

For example, consider the partial WordNet taxonomy in Figure 2.1. The Leacock-Chodorow similarity between "car" and "truck" is $-\log \frac{2}{32} = 1.20$, while the similarity between "car" and "fork" is $-\log \frac{11}{32} = 0.463$.

Resnik : (Resnik, 1995)'s approach uses WordNet and a corpus to compute the similarity between two concepts. Specifically, it computes the information content of the least common ancestor of the two concepts in the taxonomy. The Information content of a concept is calculated as the probability of encountering the concept in a large corpus.

$$sim_{Resnik}(c_1, c_2) = -\log p(lca(c_1, c_2)), \tag{2.8}$$

where $lca(c_1, c_2)$ is the least common ancestor of c_1 and c_2.

Jiang-Conrath : (Jiang and Conrath, 1997)'s measure computes the distance between concepts, therefore, the lesser the distance the more similar they are. The measure builds on Resnik's measure of similarity, but in the form of the conditional probability of encountering an instance of a child-synset given an instance of a parent

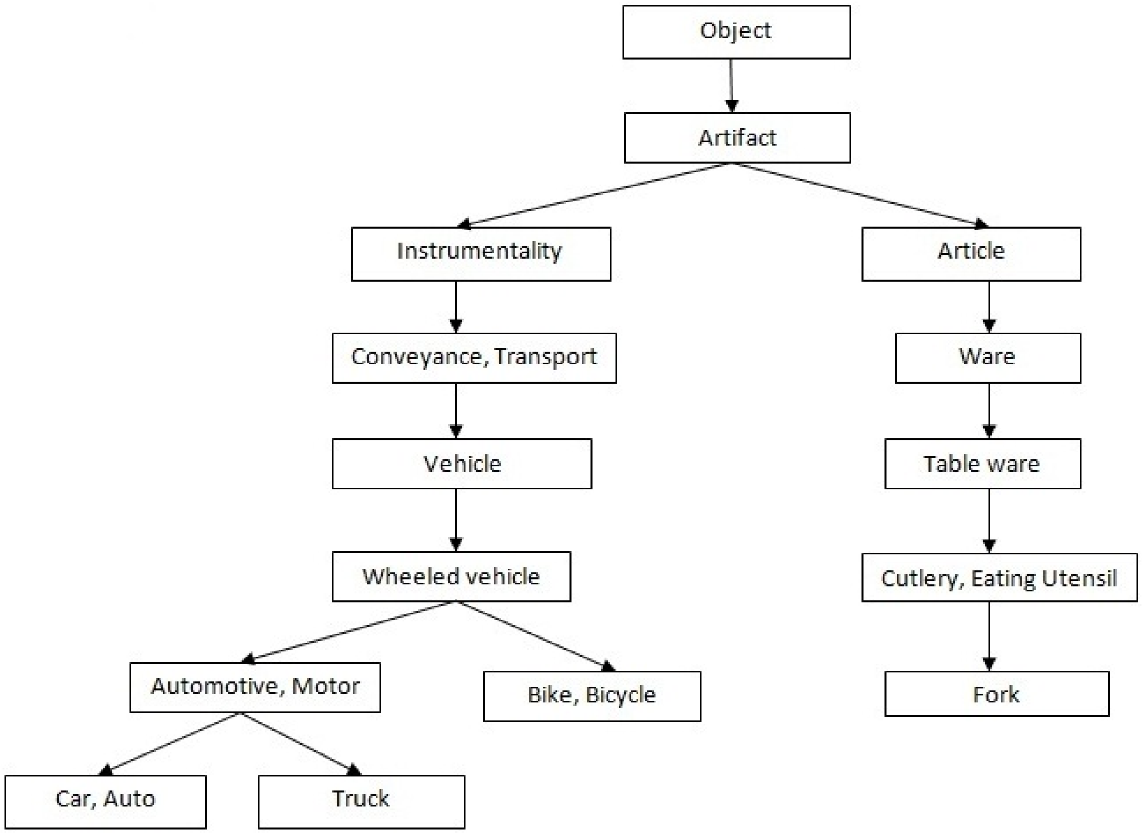


Figure 2.1: Example WordNet taxonomy for nouns

synset. The measure as shown below uses the information content of the two concepts themselves and the information content of the least common ancestor.

$$dist_J C(c_1, c_2) = 2\log(p(lca(c_1, c_2))) - \log(p(c_1)) - \log(p(c_2)) \tag{2.9}$$

Hirst-St-Onge : Most of the similarity metrics use only the hyponymy relationship in the WordNet. However, (Budanitsky and Hirst, 2006) considers all the different relations in WordNet like meronymy, hypernymy, etc. For this reason, they use the term "Semantic Relatedness" instead of semantic similarity to define what they measure. They measure semantic relatedness between two concepts as a function of the path length between the two concepts in the WordNet and the number of direction changes along the path. A path is a sequence of between two and ve links between synsets. Onnly specific patterns are allowed as part of the path. If a multi-link path between two synsets is to be indicative of some reasonable semantic proximity, the semantics of each lexical relation must be taken into consideration. Now, an upward direction corresponds to generalization (hypernymy) For example, an upward link from "apple" to "fruit" means that "fruit" is a more generalized synset than "apple". Similarly, a downward link corresponds to specialization (hyponymy). Horizontal links are less frequent than upward and downward links. Such links are usually very specic of meaning (synonymy). So, to ensure that a path corresponds to a reasonable relation between the source and the target word, they define two rules

- No other direction may precede an upward link. Once a downward or horizontal link has been used, it is not allowed to generalize again by using an upward link.

- At most one change of direction is allowed. However, it is permitted to use a horizontal link to make a transition from an upward to a downward direction.

The concepts are considered to be closely related if the path length is small and the direction doesn't change often. The semantic relatedness is computed as,

$$rel_{HS}(c_1, c_2) = C - path\_length - k * d, \tag{2.10}$$

where c_i is a concept and C, k, are constants. d is the number of direction changes along the path. If no such path exists, then the relatedness is zero. Let $C = 8$ and $k = 1$ as in (Budanitsky and Hirst, 2006)'s experiments. Then $rel_{HS}(``car'', ``truck'')$ is $8 - 2 - 1 = 5$, while $rel_{HS}(``car'', ``bike'')$ is 4.

Pointwise Mutual Information Pointwise Mutual Information (PMI) was suggested by(Turney, 2001) as an unsupervised measure of the semantic similarity between two words. It requires the existence of some large corpora like the Web or Wikipedia. Given two words w_1 and w_2, the similarity between them is defined as,

$$sim(w_1, w_2) = log_2 \frac{p(w_1 \& w_2)}{p(w_1)p(w_2)} \tag{2.11}$$

The above formula measures the degree of dependence between words w_1 and w_2. To compute the $p(w_1 \& w_2)$, Turney suggests four different type of search engine queries we can use to estimate the probability values. The best one in terms of balance between complexity and accuracy in the experimental evaluation turned out to be queries using the "NEAR" operator. The "NEAR" operator in a search engine returns documents where the words are near each other. Then the probability values are estimated as

$$p(w_1 \& w_2) \approx \frac{hits(w_1 NEAR w_2)}{WebSize} \tag{2.12}$$

$$P(w_i) \approx \frac{hits(w_i)}{WebSize} \tag{2.13}$$

2.5 Document Similarity

It is quite easy to extract useful information from highly structured text, like documents in XML format. However, extracting novel information from unstructured text documents needs many sophisticated Natural Language Processing (NLP) and Information Retrieval (IR) algorithms. At the heart of many of the NLP and IR algorithms is the need for a good similarity measure. For example, many of the algorithms for document clustering (Strehl et al., 2000), word sense disambiguation (Karov and Edelman, 1998), document classification (Goldberg et al., 2007; Calado et al., 2003), etc make use of similarity measures.

In this entire chapter, we will focus on the similarity measures and not on the algorithms for specific problems like document clustering or word sense disambiguation. Before discussing the similarity measures for documents, we need to decide how to represent documents. We discuss some of the most commonly used representations for documents below.

2.6 Document Representation

Each similarity measure assumes an underlying representation of the text. The representation defines the framework for the similarity measure. A few commonly used representations are

2.6.1 Free Text

This is the most basic representation of documents which is the surface representation (i.e. the text itself). This representation is very sparse. No information is lost because no automatic or manual transformations have been applied to the text. However, it is hard to extract information from this representation without sophisticated techniques.

2.6.2 Bag of Words

In this model, a document is represented as a collection of words without considering the ordering of the words in the document. This simple assumption helps ease the overhead of computation of similarity between two document vectors.

Assume a corpus consists of M documents: $C = \{d_1, d_2, \ldots, d_M\}$. Let the total number of terms in this corpus be N: $T = \{t_1, t_2, \ldots, t_N\}$. A common representation in this model is using a feature vector. For example, Let a $N-$ dimensional vector, $\vec{V}(d_i)$, is used to represent the feature vector for document d_i. $\vec{V}(d_i)_j$, indicates the presence of term t_j in document d_i. A common choice is to weight each element of the vector with the frequency of the corresponding term. For example, consider a corpus consisting of the following two sentences.

- D_1: John likes cars manafactured by BMW. BMW cars are powerful.
- D_2: BMW manafactures powerful cars and bikes.

The corpus consists of 11 unique terms: $T = \{$John, likes, cars, manafactured,by,BMW are, powerful, manafactures, and, bikes$\}$. The feature vector description for the first sentence is $\vec{V}(d_1) = \{1, 1, 2, 1, 1, 1, 1, 1, 0, 0, 0\}$.

2.6.3 Stemmed Representation

Stemming is one of the most natural ways to "normalize" text. For this reason, it is commonly used as a rudimentary step in many NLP and IR algorithms. Most of the stemmers can be classified as rule-based stemmers(Porter, 1997) or statistical stemmers(Krovetz, 1993). Although stemming can improve the representation of two documents for similarity comparison, it could also mislead into wrong matches. For example, using the Porter stemmer, both "marinated vegetables" and "marine vegetation" are normalized to "marin veget", which is clearly undesirable.Metzler et al. (2007) In general, the number of meaningful conversions exceeds the spurious matches which helps in computing a better similarity measure.

2.6.4 Knowledge Rich Representation

Though stemming allows us to overcome the effects of vocabulary mismatch to a certain extent, it does not handle the effects of polysemy, a single word having multiple senses. For example, two documents could have the word "bank", but could use them in very different senses. For example, the word "bank" in the first document refers to a "river bank", while the second document could use it in the sense of a "financial institution". To overcome these effects, it is useful to build representations that include contextual information. One common way of accomplishing this has been to use an external source of knowledge like the Web, Wikipedia or WordNet

2.7 Document Similarity Measures

Similarity measures between documents can be broadly classified into five different approaches based on the document representation and usage of external sources of information. The five approaches are listed below.

2.7.1 String-based approach

In this approach, documents are considered as strings and we look for approximate matches between the two strings and the degree of matching defines the similarity score. Any of the similarity measures for words discussed before can be used.

2.7.2 Vector Space Similarity

In this approach, the document is represented as a vector and the similarity is defined based on how close the two vectors are in N-dimensional space. According to the Bag of Words (BoW) model, two documents d_i and d_j can be represented as N-dimensional vectors $\vec{V}(d_i)$ and $\vec{V}(d_j)$, where $\vec{V}(D_i)_j$ is equal to the frequency of the j^{th} term in the i^{th} document. Then, any one of the vector-based similarity measures discussed before can be used to compute the similarity between the corresponding documents. Cosine Similarity is the most commonly used similarity measure since it is normalized and hence, easy to interpret. There have been a lot of variants on the Cosine Similarity based on the term weights. One of the best known methods is Term Frequency - Inverse Document Frequency (TF-IDF) (Jones, 1972) weighting. In this weighting scheme, let $\tilde{\mathbf{v}}_d = [w_{1,d}, w_{2,d}, \ldots, w_{N,d}]^T$ be the document vector for document D_d, where

$$w_{t,d} = \mathrm{tf}_t \cdot \log \frac{|D|}{|\{t \in d\}|} \tag{2.14}$$

where, tf_t is term frequency of term t in document d and $\log \frac{|D|}{|\{t \in d\}|}$ is the inverse document frequency. $|D|$ is the total number of documents in the corpus; $|\{t \in d\}|$ is the number of documents containing the term t.

2.7.3 Knowledge-based approach

This approach is mostly used for short documents. Here, in addition to the documents being compared, we assume the existence of a knowledge source like WordNet, thesauri and other such external resources to help overcome the effects of vocabulary mismatch.

For example, if a document contains the word "car", then to enrich the representation we can use the Wikipedia definition of car: "An automobile, autocar, motor car or car is a wheeled motor vehicle used for transporting passengers, which also carries its own engine or motor." This approach is mostly used for enriching very short documents which do not have much content. For example, (Metzler et al., 2007; Sahami and Heilman, 2006) use the Web to compute similarity between short texts. Consider the following two short text snippets, "UN Secretary General" and "Kofi Annan". Clearly, in this case, any similarity measure using the free text or stemmed representations will not be able to assign a non-zero similarity score. (Sahami and Heilman,

2006) use the following approach to assign a non-zero similarity score for the target example. They submit each short document as a query to a search engine and use the most frequently used words in the top 10 results as the enriched representation. Any vector based similarity measure can be used to assign a non-zero similarity score between the enriched representations.

2.7.4 Corpus-based approach

The effects of polysemy and vocabulary mismatch are handled using co-occurrence statistics to help define similarity scores between terms. The hypothesis is that two words which co-occur a lot of times (proportional to the frequency of the words themselves) are semantically similar. This approach can be further broken into approaches based on linear algebra and probabilistic approaches. Approaches based on Latent Semantic Analysis (LSA) (Deerwester et al., 1990) would fall into the linear algebra based methods while approaches based on Probabilistic Latent Semantic Analysis (PLSA) (Hofmann, 1999a), Latent Dirichlet Allocation (LDA) (Blei et al., 2003) fall into the probabilistic approaches.

Latent Semantic Analysis (LSA) LSA is a technique to map terms and documents into a concept space and is not a similarity measure. But this mapping makes simple similarity measures much more accurate by taking into account polysemy and synonymy. Synonymy and polysemy are handled by analyzing the relations between terms through the mapping. The mapping is essentially a dimensionality reduction technique using Singular Value Decomposition (SVD)(Golub and Reinsch, 1970). SVD refers to the technique of factorizing a $m \times n$ matrix X, into the following form,

$$X = U\Sigma V^T \tag{2.15}$$

where U is an $m \times m$ orthogonal matrix, the matrix Σ is an $m \times n$ diagonal matrix with non-negative real numbers along the diagonal, and V^T, an $n \times n$ orthogonal matrix.

LSA uses the term-document matrix X, of the occurrence of terms in documents. It is a sparse matrix where X_{ij} indicates the presence of term t_i in document d_j. Usually, the terms in a document are weighted using using the standard TF-IDF method. Therefore X looks like,

$$\begin{array}{cc} & \mathbf{d}_j \\ & \downarrow \\ \mathbf{t}_i^T \rightarrow & \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix} \end{array} \tag{2.16}$$

where $x_{i,j} =$ "TF-IDF weight of term t_i in document d_j. Now a row in this matrix will be a vector corresponding to a term, giving its relation to each document:

$$\mathbf{t}_i^T = \begin{bmatrix} x_{i,1} & \dots & x_{i,n} \end{bmatrix} \tag{2.17}$$

Likewise, a column in this matrix will be a vector corresponding to a document, giving its relation to each term:

$$\mathbf{d}_j = \begin{bmatrix} x_{1,j} \\ \vdots \\ x_{m,j} \end{bmatrix} \tag{2.18}$$

Now the dot product $\mathbf{t}_i^T \mathbf{t}_p$ between two term vectors gives the correlation between the terms over the documents. The matrix product XX^T contains all these dot products. $XX^T{}_{i,j}$ (which is equal to element $(XX^T{}_{j,i})$ contains the dot product $\mathbf{t}_i^T \mathbf{t}_j (= \mathbf{t}_j^T \mathbf{t}_i)$. Likewise, the matrix $X^T X$ contains the dot products between all pairs of document vectors, giving the cosine similarity between the document pairs: $\mathbf{d}_j^T \mathbf{d}_q = \mathbf{d}_q^T \mathbf{d}_j$.

Now assume that there exists a decomposition of X such that U and V are orthonormal matrices and Σ is a diagonal matrix. This is called a singular value decomposition (SVD) of X:

$$X = U \Sigma V^T \tag{2.19}$$

The matrix products giving us the term and document correlations then become

$$\begin{aligned} XX^T &= (U\Sigma V^T)(U\Sigma V^T)^T = (U\Sigma V^T)(V^{T^T} \Sigma^T U^T) = U\Sigma\Sigma^T U^T \\ X^T X &= (U\Sigma V^T)^T (U\Sigma V^T) = (V^{T^T} \Sigma^T U^T)(U\Sigma V^T) = V\Sigma^T \Sigma V^T \end{aligned} \tag{2.20}$$

Since $\Sigma\Sigma^T$ and $\Sigma^T\Sigma$ are diagonal, U must contain the eigenvectors of XX^T and V must contain the eigenvectors of $X^T X$. Both products have the same non-zero eigenvalues, given by the non-zero entries of $\Sigma\Sigma^T$.

The decomposition looks like this:

$$\begin{array}{ccccccc} & X & & & U & \Sigma & V^T \\ & (\mathbf{d}_j) & & & & & (\hat{d}_j) \\ & \downarrow & & & & & \downarrow \\ (t_i^T) \rightarrow & \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix} & = & (\hat{t}_i^T) \rightarrow & \begin{bmatrix} \begin{bmatrix} \\ u_1 \\ \\ \end{bmatrix} & \dots & \begin{bmatrix} \\ u_l \\ \\ \end{bmatrix} \end{bmatrix} & \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix} & \begin{bmatrix} \begin{bmatrix} & v_1 & \end{bmatrix} \\ \vdots \\ \begin{bmatrix} & v_l & \end{bmatrix} \end{bmatrix} \end{array} \tag{2.21}$$

The values $\sigma_1, \dots, \sigma_l$ are called the singular values, and $u_1, \dots, u_l$ and $v_1, \dots, v_l$ the left and right singular vectors. Notice how the only part of U that contributes to $\mathbf{t}_i$ is the i^{th} row. Let this row vector be called $\hat{t_i}$. Likewise, the only part of V^T that contributes to $\mathbf{d}_j$ is the j^{th} column, $\hat{d_j}$. It turns out that when you select the k largest eigenvalues, and their corresponding eigenvectors from U and V, you get the rank k approximation to X. We write this approximation as

$$X_k = U_k \Sigma_k V_k^T \tag{2.22}$$

This approximation ensures the frobenius norm (Golub and Loan, 1996) between X

and X_k is minimized . The Frobenius norm of a $m \times n$ matrix X is defined as,

$$\|X\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |X_{ij}|^2} \tag{2.23}$$

The approximation also transforms the term and document vectors into a concept space. The vector $\hat{t_i}$ then has k entries, each giving the occurrence of term i in one of the k concepts. Likewise, the vector $\hat{d_j}$ gives the relation between document j and each concept. Now we can compute the semantic similarity between two documents using cosine similarity. In the transformed concept space, we have taken advantage of correlations between terms. Even if $X_{ij} = 0$ but term t_i was semantically related to d_j, then in the transformed matrix, the (i, j) entry would be non-zero. Note that term t_i can be considered semantically related to document d_j if term t_i co-occurs a lot with other terms appearing in document d_j. There are two major disadvantages of LSA. First, the concept space is hard to explain intuitively because the rows in the transformed matrix do not correspond to actual terms appearing in the document, but instead a linear combination of the term weights. Secondly, the computational complexity of performing SVD on the original matrix is $O(N^3)$, where N is the total number of terms and documents, which makes it infeasible to apply on large data sets. Also there is no easy way to determine the optimal number of eigenvectors, k, to use for computing similarity.

2.7.5 Kernel Methods

All kernel methods can be classified under the vector-based approaches, but the large amount of literature on the Kernel methods warrants a separate section. Kernel methods are used when the underlying data, for example, graphs or text, cannot be easily represented using feature vectors for computing structural (graphs) or content similarity (text). In such cases, the data is transformed into higher dimensional space and then the inner product in the higher dimensional space is used to compute similarity, which is called as the kernel function. Mathematically, a function that calculates the inner product between mapped examples in a feature space is a kernel function, that is for any mapping

$\phi : D \rightarrow F$, a kernel function is defined as,

$$K(d_i, d_j) = \langle \phi(d_i), \phi(d_j) \rangle \tag{2.24}$$

Note that the kernel computes this inner product by implicitly mapping the examples to the feature space. The mapping ϕ transforms an n dimensional example into an N dimensional feature vector. $\phi(d) = (\phi_1(d), \ldots, \phi_N(d)) = (\phi_i(d))$ for $i = 1, \ldots, N$. For example, even the cosine similarity is a kernel function where the words in the string are the different dimensions.

The main idea of string kernels is to compare sentences based on the subsequences they contain instead of words. For example, the word “car” would match both “card” and “canary” but with different weights based on the length of the subsequence and the degree of contiguity in the match. The advantage of this approach is that it can detect words with different suffixes or prefixes. For example, the words "Computer-based", "microcomputer", "computers" have common subsequences. Of course, lots of non-related word pairs, like ("Neanderthals" , "Netherlands"), too have common subsequences, but the overall number of common subsequences between similar sentences exceed the number in dissimilar. sentences. This is because a sentence will contain more than one word, but the algorithm maps the whole sentence into one feature space: the concatenation of all the words and the spaces (ignoring the punctuation) is considered as a unique sequence. One of the most commonly used kernels is the String Subsequence Kernel (SSK)(Lodhi et al., 2002).

Let $\sum$ be a finite alphabet. A string is a finite sequence of characters from Σ, including the empty sequence. For strings s and t, let $|s|$ denote the length of the string $s = s_1 ... s_{|s|}$, with st the string obtained by concatenating the strings s and t and with $s[i : j]$ substring $s_i ... s_j$. We say that u is a subsequence of s if there exist indices $i = (i_1, ..., i_{|u|})$ with $1 \leq i_1 < ... < i_{|u|} \leq |s|$ such that $u = s[i]$. The length $l(i)$ of the subsequence in s is $i_{|u|} - i_1 + 1$. Feature mapping ϕ_u for string s is given by defining ϕ_u for each $u \in \Sigma^n$ as

$$\phi_u(s) = \sum_{i:u=s[i]} \lambda^{l(i)} \tag{2.25}$$

for some $\lambda \leq 1$, the decay factor, weighing down the subsequences with larger gaps. These features measure the number of occurrences of subsequences in the string s weighting them according to their lengths. Hence, the inner product of the feature vectors for two strings s and t gives a sum over all common subsequences weighted according to their frequency of occurrence and lengths.

$$\begin{aligned}
K_n(s,t) &= \sum_{u \in \sum^n} \phi_u(s)\phi_u(t) && (2.26)\\
& \sum_{u \in \Sigma^n} \sum_{i:u=s[i]} \sum_{j:u=t[j]} \lambda^{l(i)+l(j)} && (2.27)
\end{aligned}$$

The problem with this Kernel function is that it becomes intractable when computed naïve for $n > 4$. But it turns out that it can be computed efficiently using dynamic programming. Instead of computing $K_n(s,t)$ directly, we compute,

$$k_i^{'}(s,t) = \sum_{u \in \Sigma^n} \sum_{i:u=s[i]} \sum_{j:u=t[j]} \lambda^{|s|+|t|-i_1-j_1+2}, \quad (2.28)$$

We can now recursively compute $k_i^{'}$ and hence compute K_n.

$$\begin{aligned}
k_0^{'}(s,t) &= 1,\ for\ all\ s,t, && (2.29)\\
k_i^{'}(s,t) &= 0,\ if\ min(|s|,|t|) < i, && (2.30)\\
k_i(s,t) &= 0,\ if\ min(|s|,|t|) < i, && (2.31)\\
k_i^{'}(sx,t) &= \lambda K_i^{'}(s,t) + \sum_{j:t_j=x} k_i^{'}(s,t[1:j-1])\lambda^{|t|-j+2}, && (2.32)\\
& \quad i = 1,\ldots,n-1 \\
k_i(s,t) &= \lambda K_n(s,t) + \sum_{j:t_j=x} k_{n-1}^{'}(s,t[1:j-1])\lambda^2, && (2.33)
\end{aligned}$$

One possible variant of SSK is to use syllables or words instead of characters. One important advantage of this method is the reduction in the dimensionality of the feature space . The advantage of using syllables over words is in detecting similarities between words with same prefixes and suffixes. The problem with this approach is in breaking word into syllables.

2.8 Structured Object Similarity

Recently, there has been a surge in text being modeled using graphs, trees. The idea is that this representation is useful for many natural language processing tasks. Hence, similarity measures between such discrete data structures need to be developed. There has been considerable interest and growth in this relatively new field. For example, similarity measures have been defined for graphs, nodes in the same graph, trees, etc. Here we will go over two such similarity measures.

2.8.1 Graph Similarity

(Jeh and Widom, 2002) propose a similarity measure for any two objects which have a notion of relationship defined between them, and therefore, they are not specific to text. In Simrank, the objects are represented as nodes and object to object relationship as edges. The main idea of the algorithm is that two objects are similar if the objects they are related to are similar themselves. For example, two scientific publications are considered to be similar if they cite papers which are themselves similar and the papers they are cited by are similar. In the domain of online shopping, two users are similar if they buy similar products and two products are similar if they are bought by similar customers.

Formally, let the objects and relationships be represented by a directed graph $G = (V, E)$, where nodes in V represent the objects and the edges in E represent the relationships. Let $I(v)$ and $O(v)$ represent the set of incoming neighbors and outgoing neighbors for a vertex, respectively. Individual incoming neighbors are represented as $I_i(v)$, for $1 \leq i \leq |I(v)|$ and individual outgoing neighbors are represented as $O_i(v)$, for $1 \leq i \leq |O(v)|$. Then the formula for SimRank can be written as

$$sim(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(a), I_j(b)) \tag{2.34}$$

where C is a decay constant between 0 and 1. The contribution of any node x to the similarity between a and b reduces with the number of hops from a and b to x.

The obvious and natural base case is $sim(a,a) = 1$. In the case of a bipartite graph, we can use both the incoming edges and outgoing edges as shown below.

$$sim(a,b) = \frac{C_1}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(a), I_j(b)) \tag{2.35}$$

$$sim(c,d) = \frac{C_2}{|O(c)||O(d)|} \sum_{i=1}^{|O(c)|} \sum_{j=1}^{|O(d)|} s(O_i(c), I_j(d)) \tag{2.36}$$

Depending on the specific problem, it might make sense to use either just the incoming edges or both incoming and outgoing edges. For example, in the case of online shopping, every user has only outgoing edges, which represent which items he/she bought. Similarly, every item has only incoming edges, representing which users bought it. In this case, only incoming edges are considered for computing similarity between users. Similarly, only outgoing edges are used for computing similarity between items. Whereas, in the case of scientific papers, it makes sense to incorporate the similarity between cited papers and papers which cite the papers in question to compute similarity. In this case, a combination of the similarity values using incoming edges and outgoing edges should be used.

The algorithm for computing simrank scores is a simple iterative algorithm starting with $sim(a, a) = 0$ and then apply the recursive equations 2.36 to update the similarity values till they converge. The problem with the above formulation of similarity is that we need to compute similarities between all pairs of nodes and this makes the process computationally expensive. A simple solution to the problem is to compute the similarity between nodes which are within a distance of k and assume the similarity of a node with other nodes farther than k edges is zero. This helps speed up the algorithm and makes it feasible on reasonable large data sets.

2.8.2 Tree Kernels

Tree kernels are used for many NLP tasks which benefit from the syntactic information associated with the document. For example, part-of-speech tags and parse trees have proven to be very useful in syntactic parsing(Collins and Duffy, 2001), relation extraction(Bunescu and Mooney, 2005), named entity recognition(Cumby and Roth, 2002), semantic parsing(Moschitti, 2004), etc. The main advantage of tree kernels is the high number of rich syntactic features and the learning algorithm automatically selects the most relevant features for the task at hand. In many learning algorithms, the features are selected manually, but this requires a lot of effort. To overcome this problem, tree kernels provide a large number of features whose similarity can be computed efficiently using Kernel methods. There are two main types of tree kernels, the SubTree Kernel (STK) (Vishwanathan and Smola, 2003) and Sub-

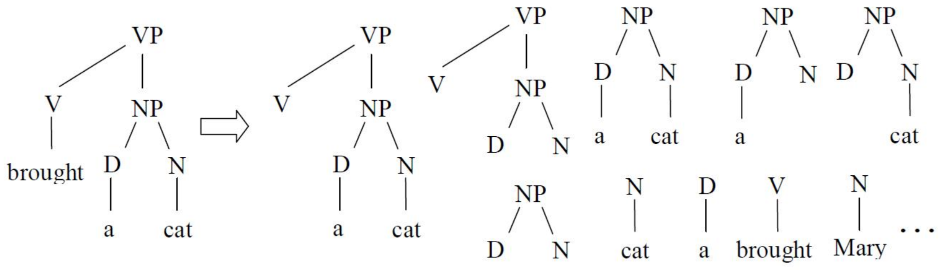


Figure 2.2: Example of Sub-Set Trees (Bloehdorn and Moschitti, 2007)

Set Tree Kernel (SSTK) (Bloehdorn and Moschitti, 2007). In this section, we will describe tree kernels with reference to syntactic parse trees. But it must be noted that the methods are general and they are not specific to parse trees. A subtree rooted at any node is defined as that node and all nodes which are descendants of that node. A sub-set tree is different from subtree in the sense that the leaves of a sub-set tree can be associated with non-terminal symbols. The SSTs satisfy the constraint that they are generated by applying the same grammatical rule set which generated the original tree. It is best explained using an example as shown in Figure 2.2.

Having defined subtrees and sub-set trees, we can now discuss how to compute similarity between two trees. Let the set of features be represented as $F = \{f_1, f_2, \dots f_m\}$, and indicator function $I_i(n)$ is equal to 1 if the target f_i is rooted at node n and 0 otherwise. The tree kernel is defined as,

$$K(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \delta(n_1, n_2), \tag{2.37}$$

where N_{T_i} is the set of nodes in tree T_i and $\delta(n_1, n_2) = \sum_{i=1}^{|F|} I_i(n_1) I_i(n_2)$. Clearly, $K(T_1, T_2)$ computes the number of common fragments (subtrees). $\delta(n_1, n_2)$ can be computed using a simple recursive procedure as shown below

1. If the edges at n_1 and n_2 are different, then $\delta(n_1, n_2) = 0$.

2. If the edges at n_1 and n_2 are same, and n_1 and n_2 have only leaf children (pre-terminal), then $\delta(n_1, n_2) = 1$

3. If the edges are same and n_1 and n_2 are not pre-terminals, then $\delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (\sigma + \delta({c^j}_{n_1}, {c^j}_{n_2})$

If $\sigma = 0$, then the above recursive procedure computes the Subtree Kernel and if $\sigma = 1$, the procedure computes the sub-set tree kernel (Collins and Duffy, 2001). The problem with the above formulation is that it is computationally expensive, taking $O(|N_{T_1}||N_{T_2}|)$. To compute it efficiently, (Moschitti, 2006) proposes computing $\delta(n_1, n_2)$ for only (n_1, n_2) pairs whose edges are identical. This set of pairs can be precomputed efficiently using a simple inorder traversal and then looking at the node pairs in a linear fashion similar to the merge algorithm in mergesort. This simple trick leads to an efficient linear time algorithm which runs in time $O(|N_{T_1}| + |N_{T_2}|)$.

2.9 Machine Learning Approaches

Machine learning approaches are used to "learn" similarity functions given training data. Given training data in the form of pairs of similar objects and dissimilar objects, the problem is to estimate parameters for a class of similarity/distance functions such that the learnt similarity/distance function satisfies the training data. For example, the Mahalanobis distance function (Mahalanobis, 1936), is used to compute the distance between two vectors x_i and x_j as,

$$d_A(x_i, x_j) = (x_i - x_j)^T A(x_i - x_j) \tag{2.38}$$

The distance function $d_A(x_i, x_j)$ is parameterized by the positive definite matrix A. Consider relationships constraining the similarity or dissimilarity between pairs of points. Two points are similar if the Mahalanobis distance between them is smaller than a given upper bound, i.e., $d_A(x_i, x_j) \leq u$ for a relatively small value of u. Similarly, two points are dissimilar if $d_A(x_i, x_j) \geq l$ for sufficiently large l. Such constraints are typically inputs for many supervised learning problems, and can also be readily inferred in a classification setting where class labels are known for each instance: distances between points in the same class can be constrained as similar, and points in different classes can be constrained as dissimilar.Typically, this learned distance function is used to improve the accuracy of a k-nearest neighbor classifier, or to incorporate semi-supervision into a distance-based clustering algorithm.

(Davis et al., 2007a) learn a Mahalanobis distance metric by solving the following optimization problem: Given pairs of similar points S and pairs of dissimilar points D, the distance metric learning problem is

$$\begin{aligned} &min_A && dist(A, A_0) && && (2.39) \\ &\text{subject to} && d_A(x_i, x_j) \leq u && (i,j) \in S && (2.40) \\ & && d_A(x_i, x_j) \geq l && (i,j) \in D && (2.41) \end{aligned}$$

where A_0 is the given initial positive definite matrix and $dist(A, A_0)$ computes the probabilistic distance between the two matrices with the assumption that both the matrices are samples of a Gaussian distribution using KL-divergence. They express the objective function as a type of Bregman divergence and use Bregman's method (Censor and Zenios, 1997) to solve the metric learning problem

2.10 Avoiding Pairwise Similarity Computation

The previous sections contain different similarity measures. However, one of the simplest operations required for most machine learning tasks like classification, collaborative filtering and clustering is computing the K-Nearest Neighbor (KNN) graph. The KNN graph is a simple graph, $G = (V, E)$ where V is the set of items and there exists an edge, $(u, v) \in E$ if and only if they are within the K most similar items of each other. However, the biggest challenge in constructing the KNN graph is the computation involved. A brute-force approach requires computing similarity between all pairs of items which can be prohibitively large for most medium to large-scale data sets.

Not surprisingly, the problem has attracted a lot of research and there are a variety of ways to avoid the pairwise computation. For example, a KNN graph can be constructed simply by repetitively invoking K-NN search for each object in the dataset. Various tree-based data structures are designed for both general metric space and Euclidean space (Beygelzimer et al., 2006; Liu et al., 2004) to avoid pairwise computation of similarity. . Locality Sensitive Hashing (LSH) (Gionis et al., 1999) is a promising method for approximate KNN search. Such hash functions have been designed for a range of different similarity measures, including hamming distance, lp with p (0, 2] (Datar and Indyk, 2004), cosine similarity (Charikar, 2002), etc.

However, the computational cost remains high for achieving accurate approximation, and designing an effective hash function for a new similarity measure is non-trivial.

(Dong et al., 2011) proposed a scalable, efficient and generalized algorithm to build KNN graphs without the need for $O(N^2)$ computations. Assume the dataset, V, has a size of N and let $\sigma : V \times V \to \mathbb{R}$ be a similarity measure. For each $v \in V$, let $B_K(v)$ be v's KNN, i.e. the K objects in V (other than v) most similar to v. For the purposes of theoretical analysis, they consider a metric space, $d : V \times V \to [0, +\inf)$. Since smaller distance means higher similarity, we simply let $\sigma = d$. For any $r \in [0, +\inf)$, the r-ball around $v \in V$ is defined as $B_r(v) = u \in V | d(u, v) \leq r$. A metric space V is said to be growth restricted if there exists a constant c, such that

$$|B_{2r}(v)| \leq c|B_r(v)|, \forall v \in V. \tag{2.42}$$

The smallest such c is called the growing constant of V ,which is a generalization of the concept of dimensionality and captures the complexity of the dataset. The algorithm is based on the following simple principle, a neighbor of a neighbor is very likely to be a neighbor. Suppose there exists an approximate nearest-neighbor graph, G, then the approximation can be improved by exploring the neighbors of neighbors for every item.

The algorithm begins by initializing a random KNN graph. Then the approximate graph is refined iteratively. In each iteration, for each object, the nearest neighbor list is updated by computing the similarity between the object and neighbors of neighbors. The algorithm is terminated when there are no updates performed in the previous iteration. They report the empirical complexity of the algorithm as

$O(n^1.14)$ when tested on the corel data set (662,317 objects) (Frank and Asuncion, 2010) and the DBLP data set (857,820 objects).

2.11 Evaluation

There are two ways of evaluating similarity measures, intrinsic and extrinsic. In an intrinsic evaluation, similarity scores between the objects in context are obtained from multiple human experts. The correlation between these similarity scores and the scores outputted by the similarity measure is often used as an evaluation measure. For example, (Rubenstein and Goodenough, 1965) obtained synonymy judgments of 51 human subjects on 65 pairs of words. The pairs ranged from “highly synonymous” (gemjewel) to “semantically unrelated” (noonstring). Subjects were asked to rate them on the scale of 0.0 to 4.0 according to their “similarity of meaning” and ignoring any other observed semantic relationships (such as in the pair “journeycar”). (Miller and Charles, 1991) subsequently extracted 30 pairs from the original 65, taking 10 from the “high level (between 3.0 and 4.0), 10 from the intermediate level (between 1 and 3), and 10 from the low level (0 to 1) of semantic similarity”, and then obtained similarity judgments from 38 subjects. (Budanitsky, 2001) compare the different similarity measures for words based on WordNet taxonomy using the correlation between the similarity score obtained by similarity measure and the score given by human experts. Table 2.1 shows the results of the comparison for the described similarity measures.

One of the major criticisms against this approach is it is often very hard to obtain similarity scores for a large number of object pairs. In an extrinsic evaluation, the

Similarity Measure	**M&C**	**R&G**
Hirst and St-Onge	.744	.786
Leacock and Chodorow	.816	.838
Resnik	.774	.779
Jiang and Conrath	.850	.781

Table 2.1: The coefficients of correlation between human ratings of similarity (by Miller and Charles and by Rubenstein and Goodenough) and the five computational measures.

similarity scores are used in an external task and the performance in the task is used as the evaluation metric of the similarity measure. A commonly used approach to evaluate a word similarity measure is to use the similarity measure to find synonyms in a standardized test like TOEFL. The most similar word among the given choices is chosen as the synonym. For example, (Turney, 2001) evaluated the similarity measure using PMI on the TOEFL synonymy tests and obtained a score of 72.5%, which is higher than the accuracy obtained by LSA(64.4%) and the average non-English college applicant(64.5%).

One of the commonly used tasks for extrinsically evaluating similarity measures is Clustering. Clustering is the task of grouping similar data points into groups (clusters). A common approach is to represent the data using a graph, $G = (V, E)$, where the set of nodes, V is used to represent data points. Usually the set of edges, E, represents the similarity edges between the nodes and the edges are weighted using a function, $w(u, v) \rightarrow \mathbb{R} : (u, v) \in E$. Mathematically, the clustering task is ,given a graph: $G = (V, E)$ and the number of clusters k, to find a partition function $P(V) = \{C_1, C_2, \ldots, C_k\}$ such that it minimizes an objective function defined over the partition function. In general, an objective function for clustering is minimized when points within each cluster are more similar to each other than to points across clusters. A commonly used objective function is normalized cut (Kulis et al., 2009),

$$\Omega(C_1, C_2, \ldots, C_k) = \sum_{i=1}^{k} \sum_{u \in C_i, v \in V - C_i} \frac{w(u, v)}{\sum_{w \in V - u} w(u, w)} \tag{2.43}$$

When there is no ground truth information about the clusters, the value of the objective function can be used as the evaluation measure for the clustering. However, when there is ground truth information available, we use Normalized Mutual Information as the measure of clustering accuracy. Mutual Information is a symmetric measure which quantifies the statistical information between two distributions. Let $I(X, Y)$ denote the amount of mutual information between the distributions X and Y. Since I(X,Y) has a lower bound of zero and no upper bound, we normalize the quantity using the entropy of X and Y. Thus the normalized mutual information used is,

$$NMI(X, Y) = \frac{I(X, Y)}{\sqrt{H(X)H(Y)}} \tag{2.44}$$

We estimate NMI using the samples provided by the clusterings. Let n be the total number of objects (nodes) to be clustered. Let $k(A)$ be the number of clusters according to clustering A and let $k(B)$ be the number of clusters according to clustering B. Let n_h^A denote the number of objects in cluster C_h according to clustering A , and let n_l^B denote the number of objects in cluster C_l according to clustering B . Let $n_{h,l}$ denote the number of objects that are in cluster C_h according to algorithm A as well as in group C_l according to clustering B. Then, NMI is estimated according to 3.46

$$\widehat{NMI(A,B)} = \frac{\sum_{h=1}^{k(A)} \sum_{l=1}^{k(B)} n_{h,l} log(\frac{n.n_{h,l}}{n_h^A n_l^B})}{\sqrt{\sum_{h=1}^{k(A)} \frac{n_h^A}{n} \sum_{h=1}^{k(B)} \frac{n_h^B}{n}}} \tag{2.45}$$

This evaluation measure has some nice properties such as $NMI(X,X) = 1$. Also $\widehat{NMI(X,Y)} = 1$ if the clusterings X and Y are the same except for a permutation of the cluster labels. $NMI(X,Y) = 0$ if the clustering X is random with respect to Y or vice versa.

Therefore, we compare the clusterings obtained by an algorithm X against the clustering Y, induced by the ground truth clusters using NMI. The higher the value the better the clustering obtained. (Strehl et al., 2000) compares 5 different similarity measures, including cosine similarity and Jaccard index using clustering as the external task. They perform the clustering using weighted graph partitioning on the Yahoo! industry web pages (966 web pages partitioned manually into 10 clusters) and Yahoo! news data set (2340 documents partitioned manually into 20 clusters) (Craven et al., 1998). They use NMI as the evaluation measure and find that cosine similarity performs the best with NMI score of 0.192 on the Yahoo! industry data and 0.240 on the Yahoo! news data set.

Another commonly used task to compare similarity measures is classification. Classification accuracy is used to evaluate the performance of a similarity measure on a classification task. For example, (Saunders et al., 2002) compares three different variants of the string kernel and cosine similarity using TF-IDF weighting on a classification data set. The variation is due to the difference in feature mapping. The three variants use characters, words and syllables as features with $n = 3$. They perform the classification on the acq-category of the Reuters-21578 collection. The acq-category consists of 1000 documents. They use SVM as the classifier in conjunction with the kernel methods. Classification error is used as the evaluation metric. The syllable kernel performs the best among the string kernels with a classification error of 12.29% while the cosine similarity kernel performs the best with a classification error of 8.00%.

CHAPTER III

Integrating Mulitple Similarity Measures

3.1 Introduction

Nowadays, relational data are universal and have a broad appeal in many different application domains. Similarity graphs are estimated from data and may not accurately reflect the semantic relationship between the objects. It is trivially clear that the more accurate the similarity graph reflects the true semantic relationship better the performance of tasks like classification, clustering and other data mining tasks.

There exists many different similarity measures for text. Refer to II for a discussion of the different similarity measures.

But currently, almost all data can be represented using much more than text. For example, publications have many heterogeneous features like text, citations, authorship information, publication venue information, etc. In most cases, similarity is computed using just one feature type or similarity is computed using two feature types individually and then combined in a linear weighted fashion.

In the case of publications, text is the most commonly used feature type for computing similarity. For example, (Hassan and Radev, 2010) use both citations and text to compute similarity between publications . They compute textual similarity using cosine similarity. Two papers are considered to be similar if one paper cites the other. The authors combine the similarity between the two papers due to citation (CS) and text (TS) in a linear fashion as follows

$$S(x, y) = \alpha CS(x, y) + (1 - \alpha)TS(x, y) \tag{3.1}$$

where, $\alpha \leq 1$ and $S(x, y)$ represents the combined similarity between publications x and y.

This formulation clearly does not take into account any dependency between the different feature spaces (in this case, text and citation space).

Rocklin and Pinar (2011) proposes multi-type edges to represent similarity due to different spaces. They use a graph to represent the objects and multiple edges between any pair of nodes. Each edge represents similarity due to a particular feature type. For example, there can be two edges between a pair of nodes, where the first edge corresponds to similarity due to text while the second edge corresponds to similarity due to citations. They propose algorithms to find how to aggregate the similarities linearly so that the resulting graph can be clustered well. They measure the quality of clustering using modularity (Clauset et al., 2004). In other words, they propose algorithms to learn α in 3.1. Thus, it does not learn across feature spaces.

Even in the case of the same feature space, higher order dependencies are not taken into account. For example, let publication X contain two words $\{w_1, w_2\}$ and publication Y contain $\{w_2, w_3\}$ then this indicates that words w_1 and w_3 are similar to each other and should affect the similarity between two other publications containing them.

Nevertheless, (Ganiz et al., 2009) takes these dependencies into account and represents each document in the classical vector space model. They use a Naive Bayes classifier to classify text documents using an augmented representation of the documents. The new representation includes the words present in the document as well as words which are present in higher order dependencies. For example, in the classical vector space model, publication X's vector representation would include only w_1 and w_2, but not w_3. Whereas, in the new representation all the words are weighted by the number of higher order paths which contain the word. They demonstrate that including higher order dependencies improves classification accuracy and most of the information required for classification is contained in second order dependencies.

Let dataset D contain N documents belonging to classes $\{c_1, c_2, \ldots, c_K\}$. Let the set of documents belonging to class c_i be represented as D_i. If document d_l contains terms $\{w_i, w_k\}$ and document d_r contains terms $\{w_k, w_j\}$ then $w_i - d_l - w_k - d_r - w_j$ represents a higher order path in dataset D. Let $\psi(w_i, D)$ denote the number of higher-order paths that contain term w_i given the dataset D, and let $\Phi(D)$ denote the total number of higher-order paths in D. The parameter estimation equations of the Naive Bayes classifier are:

$$\hat{P}(w_i|c_j) = \frac{1 + \psi(w_i, D_j)}{2 + \Phi(D_j)} \tag{3.2}$$

and

$$\hat{P}(c_j) = \frac{\Phi(D_j)}{\sum_{i=1}^{K} \Phi(D_i)} \tag{3.3}$$

(Kontostathis and Pottenger, 2006) gives a mathematical proof for the dependence of Latent Semantic Indexing (LSI)(Deerwester et al., 1990) on higher order dependencies. If two documents X and Y have an initial similarity of zero (before applying LSI) and if the similarity becomes non-zero after using LSI, then it implies that there exists a higher order dependency between the two documents. Higher order co-occurrences have also been used earlier in word sense disambiguation (Schütze, 1998) and stemming (Xu and Croft, 1998)

Although, there has been some work on using higher order dependencies it is not easily extended for multiple feature spaces, especially for heterogeneous feature types like citations, text, etc. In the case of using citations and text, it is not immediately clear how to model higher order dependencies when using multiple feature types.

One possible approach is to represent all the objects and features in a single graph. For example, in the case of publications, let $G = (V, E)$ be used to represent the publications and keywords. The set of nodes $V = \{p_1, p_2, \ldots, p_n, k_1, \ldots, k_m\}$ represents the publications and keywords. There is an edge $(p_i, p_j) \in E$ if p_i cites p_j. There is an edge (p_i, k_l) if publication p_i contains keyword k_l. This approach is equivalent to merging multiple classifiers (each using a different feature type/view) into a single classifier using all the feature types which is not recommended. There could be data points where each of the classifiers is "authoritative" (high confidence) on but by merging them this confidence is diluted. (Dasgupta et al., 2001) prove that a classifier has low generalization error if it agrees on unlabeled data with a second classifier based on different "view" of the data. This justifies the search for multiple feature types to be represented separately.

(Abney, 2002) proposes a greedy algorithm to search for classifiers which have provable advantages over the classifiers found using the co-training algorithm. The greedy algorithm searches for complex classifiers which are essentially a list of atomic rules **H**. Each atomic rule predicts a label l based on the presence of a single feature h. Each atomic rule's prediction gets one vote and the classifer's output is the label l that receives the most votes. In case of a tie, there is no prediction. However, even this approach does not take care of dependencies across feature types and higher-order dependencies between the same feature type. Our algorithm seeks to estimate similarity between the objects and features by integrating all available sources of similarity.

3.2 Problem Motivation

We motivate the need for a similarity measure which uses different feature types using an example. In this example, we consider only two different feature types, but the arguments naturally extend in the case of more than two feature types. Consider the problem of multi-class classification of publications by research area. Let there be three publications, P_1, P_2, P_3 in the area of Machine Translation. Let the publications contain two keywords, $k_1 =$" Statistical Machine Translation" and $k_2 =$"Bilingual Corpora". Specifically, publications P_1 contains k_1, P_2 and P_3 contain k_2. Although the two keywords are enough for a human to clearly see that the two papers are on Machine Translation, no textual similarity measure (without using external knowledge sources) will assign a non-zero similarity value between any pair of publications.

However, there are other sources of similarity that can be useful in this case. It is likely that two papers in the same area are similar due to the structure of the citation graph. For example, it is possible that P_1 and P_2 are both cited by a lot of papers. Assume that P_1 and P_2 are similar due to citation information. This information can be used to assign a small amount of similarity between the keywords contained in the two papers. Thus, keywords k_1 and k_2 have a non-zero similarity value. Once again, this information can be used to induce a small amount of similarity between P_1 and P_3. Therefore, two features which co-occur a lot are similar. Two objects which have a lot of similar features are defined to be similar. Thus, we can learn to estimate similarity in one feature space using similarity estimates from other feature space. We refer to this type of learning as learning across feature types.

In the next two sections, we formalize and exploit this observation to improve similarity in one feature space using other feature spaces in a principled way.

3.3 Problem Formulation

We assume that the data to be analyzed consists of many heterogeneous feature types. The objects are represented by many heterogeneous feature types and an initial coarse similarity measure between different feature types as well as objects is estimated from the data. The estimated similarity values are represented using multiple layers of graphs, where each graph corresponds to a particular feature type.

We now formally define the relevant concepts,

Definition 1. Objects and Feature Types: we have a set of N objects, $O = \{o_1, o_2, \ldots, o_N\}$. Each object is represented using m different feature types, $F = \{F_1, F_2, \ldots, F_m\}$. Each feature type, F_i consists of a set of features, $F_i = \{f_{1i}, f_{2i}, \ldots, f_{n_i i}\}$. Let the set of features used to describe object o_i be represented as $FV(o_i)$

Definition 2. Object Graph: The object graph, $G_0 = \{V_0, E_0\}$, consists of the set of objects as nodes and the edge weights represent the initial similarity between the objects.

Definition 3. Feature Graphs: For each feature type F_i, we create a feature graph, $G_i = (V_i, E_i)$. The graph consists of features F_i as nodes, i.e, $V_i = F_i$. The edge weights represent the similarity between features. In the case of feature types like citations/links, the set of nodes in the graph is V_0 and similarity can be estimated using the citation/link graph structure using node similarity measures explained in Chapter II.

The object graph and the feature graphs can be viewed as different **layers** of graphs. Note that each layer of graph can be independently used for exploiting higher-order dependencies. For example, let $G_k = (V_k, E_k)$ be used to represent a keyword feature graph, where $V_k = \{k_1, k_2, \ldots, k_m)$ denotes the set of keywords and the edge weight between k_i and k_j represents similarity between the two keywords which can be used to infer similarity between two publications which contain the keywords, even though they do not share the same vocabulary. These layers are connected using the concept of layer connectivity as defined below,

Definition 4. Layer Connectivity: The two graphs, G_i and G_j, $i \neq j$ are connected as follows. There exists an edge between f_{ij} and f_{kl} if they co-occur in some object's feature vector representation. Let Z refer to the layer connectivity function.

$$Z_{f_{ij},f_{kl}} = \begin{cases} 1 & \text{if } \exists m : f_{ij} \in FV(o_m) \text{ and } f_{kl} \in FV(o_m) \\ 0 & \text{else} \end{cases}$$

Thus we have a set of heterogeneous graphs each with an initial similarity measure. We improve the similarity values in each graph layer using information from all other layers. Specifically, we incorporate higher order dependencies in the same feature type as well as dependencies across feature types to improve the initial similarity estimates.

We define two features, f_{ik} and f_{jk} of the feature type F_k to be similar if

- Two objects which contain both the features are similar, i.e, o_x contains f_{ik}, o_y contains f_{jk} and o_x and o_y are similar.

- Two features of a different type which co-occur with these features are similar, i.e, f_{xl} co-occurs with f_{ik}, f_{yl} co-occurs with f_{jk} and f_{xl} and f_{yl} are similar.

The problem is to obtain improved similarity estimates in all feature spaces. Figure 3.1 shows an example representation for objects with heterogeneous features. In this example, the objects are documents with keywords features and authorship information as another feature. Nodes of the same type constitute a "layer" of the graph and are separated by dotted lines. The edges between nodes of the same layer correspond to initial similarity weights while inter-layer edge weights correspond to feature weights.

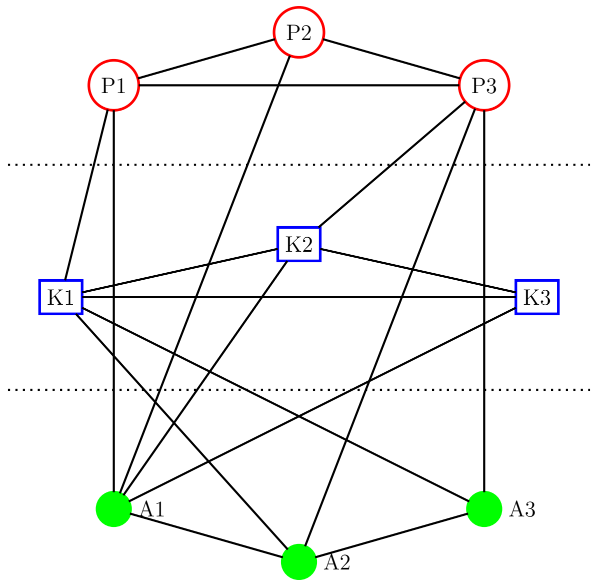


Figure 3.1: An example graph: Proposed representation for representing objects with heteregeneous features. For example, P1, P2, P3 are documents and K1, K2 and K3 are keywords and A1, A2, A3 represent authors. Objects of the same type constitute a layer of graph. The different layers are separated by dotted lines. Intra-layer edges represent initial similarity between the objects while inter-layer edges represent feature weights. For example the edge between P1 and P2 is initialized to the similarity between them while the edge between P1 and K2 represents the feature weight of K2 in P1's feature vector representation

3.4 Regularization Framework

Our approach for incorporating information from different heterogeneous feature types is inspired by the standard regularization framework for semi-supervised classification using label propagation (Mei et al., 2008; Goldberg et al., 2007). In this framework, we define a single graph $G = (V, E, w)$. The set of nodes is represented as $V = \{x_1, x_2, \ldots, x_n\}$. The edge weights w_{ij} represent similarity between two nodes i and j. Assume there exists a set of labeled nodes $L \subset V$ whose label is given by $y(x)$. The problem is to classify all other nodes using a discriminant function $f : X \to \mathbb{R}$. To compute f, (Zhu et al., 2003b) define an objective function as follows

$$\Omega(f) = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f(\mathbf{x_i}) - f(\mathbf{x_j}))^2 + \mu \sum_{x \in L} (f(\mathbf{x}) - y(\mathbf{x}))^2 \tag{3.4}$$

Essentially, the first term tries to regularize the node labeling over the network such that similar nodes get similar labels while the second term tries to minimize the difference between the true labels and the predicted labels. The first term can be written in quadratic form as shown below,

$$\Omega(f) = \frac{1}{2}\mathbf{f}^T \mathcal{L} \mathbf{f} + \mu \sum_{x \in L} (f(\mathbf{x}) - y(\mathbf{x}))^2 \tag{3.5}$$

where $\mathbf{f} = (f(\mathbf{x_1}, f(\mathbf{x_2}, \ldots, f(\mathbf{x_n})$ and $\mathcal{L}$ is the combinatorial graph Laplacian matrix, defined as $\mathcal{L} = D - W$ where D is a diagonal matrix with $d_{ii} = \sum_{j=1}^{n} w_{ij}$.

One reasonable approach to combine the different sources of similarity for performing classification is to combine the graph Laplacians in a convex fashion. For this purpose, we create m different graphs $G_1, G_2, \ldots, G_m$, where each graph, G_i consists of the set of objects as nodes and edge weights represent similarity computed using only feature type F_i. Let the Laplacian corresponding to graph G_i be represented as $\mathcal{L}_i$. Then we can perform regularization on all the graphs simultaneously as

$$\Omega(f) = \frac{1}{2}\sum_{i=1}^{m} \alpha_i \mathbf{f}^T \mathcal{L}_i \mathbf{f} + \mu \sum_{x \in L} (f(\mathbf{x}) - y(\mathbf{x}))^2 \tag{3.6}$$

where $\sum_i \alpha_i = 1$. This is equivalent to a convex combination of the similarity estimates due to different feature types as in Equation 3.1. Therefore, even this formulation does not exploit higher order dependencies in the same feature space. There are a few problems with the above framework in the context of our problem setting

- The above framework does not take into account the edges introduced by layer connectivity and hence does not use all the graphs *together*.
- The regularization is defined over nodes in the graph, whereas for improving similarity estimates we want to define a regularizer over edge weights which can be used for estimating similarity between objects using similarity information between feature types.

We define a novel regularization framework over edge weights of multiple graphs for estimating similarity between objects using similarity estimates between features and vice-versa. Let $\mathcal{W}$ refer to the similarity measure to be computed between the nodes in all the graphs, i.e, $w_{f_{ik}, f_{jk}}$ gives the similarity between features f_{ik} and f_{jk} while w^*_{ij} refers to the initial similarity value. Let $V = V_0 \cup F_1 \cup F_2 \ldots F_m$ refer to the set of vertices in all the graphs and $E = E_0 \cup E_1 \ldots E_m$ refer to the set of edges in all the graphs. The objective function is defined as

$$\Omega(w) = \alpha_0 \sum_{u,v \in V} (w_{u,v} - w^*_{u,v})^2 + \sum_{i,j=0}^{m} \alpha_{ij} \mathcal{J}(V_i, V_j) \tag{3.7}$$

where

$$\mathcal{J}(V_i, V_j) = \sum_{u_1,v_1 \in V_i} \sum_{u_2,v_2 \in V_j} Z_{u_1,u_2} Z_{v_1,v_2} (w_{u_1,v_1} - w_{u_2,v_2})^2 \tag{3.8}$$

and $\alpha_0 + \sum_{i,j=0}^{m} \alpha_{ij} = 1$

The optimization problem is to minimize the objective function. The objective function consists of two parts. The first part ensures the optimized similarity values remain close to the initial similarity values while the second term seeks to minimize the dissimilarity between all the graphs. The second term directly models our intuition of features co-occurring with other features being similar. The parameter α_0 is used to control the trade-off between the two terms. Note that if $\alpha_0 = 1$ then the minimal solution is the initial similarity itself.

The significance of the second term is explained using a simple example. Consider two graphs, G_1 and G_2. Let G_1 be the graph containing publications as nodes and edge weights representing similarity due to citation graph. Let G_2 be the graph corresponding to keywords and edge weights represent similarity between keywords. There is an edge from a node u_1 in G_1 to a node v_1 in G_2 if the publication corresponding to u_1 contains the keyword corresponding to v_1. According to this example, minimizing the objective function essentially means updating similarity values between two keywords in proportion to similarity values between the papers they are contained in and vice versa. Therefore, even though keywords like *Machine Translation* and *Word Alignment* are not similar according to textual similarity measures, they are assigned a similarity value if two papers which contain them are similar because of citation similarity.

3.4.1 Convex Optimization - Direct Solution

The objective function is a convex sum of squared differences. Hence, the objective function is a convex function in M variables where M is the total number of edges, i.e, $M = \sum_{i=1}^{m} |E_i|$. We can also add the constraint $\sum_v W_{u,v} = 1$ for normalization. Then the constrained convex optimization problem can be expressed in the form shown below,

$$\begin{aligned} &\text{minimize } \frac{1}{2}x^T P x + Qx + r \\ &\textit{subject to } Ax = b \end{aligned} \tag{3.9}$$

where x is a $M \times 1$ vector representing the edge weights along each dimension of the vector. The Karush-Kuhn-Tucker (KKT) conditions for the above problem are (Boyd and Vandenberghe, 2004),

$$Ax^* = b,\ Px^* + q + A^T v^* = 0, \tag{3.10}$$

This can be written as

$$\begin{bmatrix} P & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x^* \\ v^* \end{bmatrix} = \begin{bmatrix} -q \\ b \end{bmatrix} \tag{3.11}$$

In the above equation, v^* refers to the dual solution while x^* refers to the actual solution. Hence, we can directly solve the optimization function to obtain the similarity values by solving the above system of linear equations. Unfortunately, solving that requires inverting a $2M * 2M$ matrix which has a computationally complexity of $O(M^3)$. This is prohibitively expensive in terms of computational complexity for even moderate size data sets. Hence we need to find alternate optimization methods to minimize the objective function.

3.4.2 An Efficient Algorithm

Although the direct optimization is prohibitively expensive, we can solve it using approximate methods such as coordinate descent(Luo and Tseng, 1992). In this method, we assume all other dimensions are fixed except one and perform a descent along the chosen dimension iteratively. In the context of our problem, we assume all edge weights are fixed except one edge weight and solve the minimization of the function. For example, the partial derivative of the objective function with respect to the edge, $(u_j, v_k) \in V_i$ is shown below,

$$\frac{\partial \Omega(w)}{\partial w_{u_j,v_k}} = 2\alpha_0(w_{u_j,v_k} - w^*_{u_j,v_k}) \\ + 2 \sum_{l=0,1,\ldots,i-1,i+1,\ldots,m} \alpha_{il} \times \\ \sum_{u_2,v_2 \in V_l} Z_{u_j,u_2} Z_{v_k,v_2}(w_{u_j,v_k} - w_{u_2,v_2}) \tag{3.12}$$

To perform the descent along the edge $(u_j, v_k) \in V_i$, we set the above partial derivative to zero which gives us the following expression,

$$w_{u_j,v_k} = \frac{1}{C}\alpha_0 w^*_{u_j,v_k} + \\ + \sum_{l=0,1,\ldots,i-1,i+1,\ldots,m} \alpha_{il} \sum_{u_2,v_2 \in V_l} Z_{u_j,u_2} Z_{v_k,v_2} w_{u_2,v_2} \tag{3.13}$$

$$\tag{3.14}$$

where C is defined as

$$C = \alpha_0 + \\ + \sum_{l=0,1,\ldots,i-1,i+1,\ldots,m} \alpha_{il} \sum_{u_2,v_2 \in V_l} Z_{u_j,u_2} Z_{v_k,v_2} \tag{3.15}$$

$$\tag{3.16}$$

It can be seen that if the original similarity values are bounded in the range $[0, 1]$, then they will always remain bounded with C playing the role of a normalization constant. Note that C is a constant for a given pair of vertices and hence, C can be incorporated in w^* and w to obtain the following equation

$$\begin{aligned} w_{u_j,v_k} &= \alpha_0 w^*_{u_j,v_k} + \\ &+ \sum_{l=0,1,\ldots,i-1,i+1,\ldots,m} \alpha_{il} \sum_{u_2,v_2 \in V_l} Z_{u_j,u_2} Z_{v_k,v_2} w_{u_2,v_2} \end{aligned} \tag{3.17}$$

If we represent the initial edge weights of graph G_i by matrix W_i^* and Z_ij represent the layer connectivity matrices for graphs, G_i and G_j, then the algorithm can be written in terms of iterative matrix equations as follows,

$$W_i^t = \alpha_0 W_i^* + \sum_{l=0,1,\ldots,i-1,i+1,\ldots,m} Z_{il} W_l^{t-1} {Z_{il}}^T \tag{3.18}$$

where W_i^t refers to the matrix W_i at the end of the t^{th} iteration and ${Z_{il}}^T$ is the transpose of Z_{il}. Now, we can iteratively apply the above equation to minimize the objective function. We stop iterating when,

$$|w_{u,v}^t - w_{u,v}^{t-1}| \leq \epsilon \forall (u,v) \in E \tag{3.19}$$

where ϵ is a small threshold. We set $\epsilon = 0.01$ in all our experiments. We refer to this Similarity measure using MUltiple Graphs as SMUG.

For example, in the context of publications, let T represent the similarity between keywords and P represent the similarity between publications due to the citation graph. We combine the two different sources of similarity using the following two iterative equations

$$T_t = \alpha * T^* + (1-\alpha) * (Z.P^{t-1}.Z^T)$$
$$P_t = \alpha * P^* + (1-\alpha) * (Z^T.kT_{t-1}.Z) \tag{3.20}$$

3.4.3 Proof of Convergence

We can prove that equations 3.20 converge as the number of iterations tends to infinity.

Theorem III.1.
$$W_t - W_{t-1} \xrightarrow{t\to\infty} 0 \tag{3.21}$$

where W_t is the similarity matrix at the end of the t^{th} iteration.

Proof: In order to prove convergence, we need to show the following

$$W_t - W_{t-1} \xrightarrow{t\to\infty} 0 \tag{3.22}$$

$$W_t - W_{t-1} = (1-\alpha)Z(P_{t-1} - P_{t-2})Z^T \tag{3.23}$$
$$P_{t-1} = \alpha P_0 + (1-\alpha)Z^T W_{t-2} Z \tag{3.24}$$
$$P_{t-2} = \alpha P_0 + (1-\alpha)Z^T W_{t-3} Z \tag{3.25}$$

Thus,

$$P_{t-1} - P_{t-2} = (1-\alpha)Z^T(W_{t-2} - W_{t-3})Z \tag{3.26}$$

Substituting this in equation 3.26,

$$\begin{aligned} W_t - W_{t-1} &= (1-\alpha)^2 ZZ^T(W_{t-2} - W_{t-3})ZZ^T && (3.27)\\ &= (1-\alpha)^4 (ZZ^T)^2(W_{t-4} - W_{t-5})(ZZ^T)^2 && (3.28)\\ &= \vdots && (3.29)\\ &= (1-\alpha)^t (ZZ^T)^{\lfloor \frac{t}{2} \rfloor}(W_1 - W_0)(ZZ^T)^{\lfloor \frac{t}{2} \rfloor} && (3.30) \end{aligned}$$

In the above equation, since $\alpha < 1$ we can claim that $W_t - W_{t-1} = 0$ as $t \to \infty$ if none of the matrix terms tend to ∞. $W_1 - W_0$ is a constant matrix and ZZ^T is a transition probability matrix, where $ZZ^T(i,j)$ is the probability of random walk of length starting at i and ending at j of length 2 . Hence $(ZZ^T)^{\frac{t}{2}}$ converges to the stationary probability distribution as $t \to \infty$. Hence, $W_t - W_{t-1} = 0$ as $t \to \infty$. □

3.4.4 Layered Random Walk

The above algorithm, SMUG, has a nice intuitive interpretation in terms of random walks over different layers of the graph assuming the initial similarity weights are transition probability values and Z_{ij} matrices are normalized so that each row sums to 1. Then the similarity between two nodes, u and v in layer i, is computed as the sum of two parts. The first part is α_0 times the original edge weight. This is necessary so that the newly computed similarity estimates (after convergence) are not too far away from the initial estimates. The second part is a sum of $m - 1$ different terms. The j^{th} term is weighted by α_{ij} and is the probability of starting a random walk from u and moving to layer j for a random walk of length 1 and then returning back to vertex v in layer i. Thus, the j^{th} term is the probability of a random walk of length 3 with both intermediate vertices of the walk belonging to graph layer j.

We can see that SMUG exploits the dependencies across feature spaces to improve similarity estimates. Consider the example of publication classification mentioned in section 3.2. When we estimate similarity between the keywords in the keyword feature layer, we perform a random walk over the citation feature layer. Therefore, the similarity between the keywords k_1 and k_2 will be incremented by a value directly proportional to the similarity between the publications due to citations. Note that second and higher order dependencies are also taken into account by SMUG. That is, two papers may become similar because they contain two keywords which are connected by a path in the keyword feature layer, whose length is greater than 1. This is due to the iterative nature of the algorithm. For example, consider a set of semantically similar keywords $K = \{k_1, k_2, \ldots, k_n\}$. Assume that the initial keyword similarity estimated the similarity between these keywords is 0. Then during the first iteration, k_1 and k_2 could become similar because they are contained in publications P_1 and P_2 (similar due to citation similarity). Then k_2 could become similar to k_3 because of another pair of similar papers which contain the keywords. In this fashion, the similarity between all the keywords in the set are improved to a non-zero similarity value.

The α_{ij} values are used to control the user's knowledge about the dependency between features. For example, it is reasonable to assume that there is relatively low dependency between the venue information and authorship information. In this case, we set the corresponding value of α_{ij} to be low. This shows that we model a large number of dependencies between different feature spaces using the rich representation using heterogeneous graphs which are interconnected.

3.5 Experiments

It is very hard to evaluate similarity measures in isolation. Thus, most of the algorithms to compute similarity scores are evaluated extrinsically, i.e, the similarity scores are used for an external task like clustering or classification and the performance in the external task is used as the performance measure for the similarity scores. This also helps demonstrate the different applications of the computed similarity measure. Thus, we perform three experiments on standard benchmark data sets to illustrate the improved performance of SMUG.

3.5.1 Experiment I

We compare our similarity measure against other similarity measures in the context of classification. We also compare against a state of the art classification algorithm which uses multiple graphs to represent multiple similarity measures.

Specifically, we compare SMUG against three other similarity baselines in the context of classification which are listed below.

- *Content Similarity*: Similarity is computed using just the feature vector representation using just the text. We use cosine similarity after preprocessing each document into a tf.idf vector for the AAN data set (Radev et al., 2009b). For all other data sets, we use the cosine similarity on the feature vector representation that is available.

- *Link Similarity*: Similarity is computed using only the links (citations, in the case of publications). To compute link similarity, we use a node similarity algorithm (Harel and Koren, 2001). The algorithm computes similarity between nodes in the citation graph using a random walk of length 3. Table 3.1 lists the top 10 similar pairs of papers according to the computed link similarity.

- *Linear combination*: The content similarity and link similarity are combined in a linear fashion as shown in equation 3.1. We report the performance of the similarity measure parameterized by α.

We use a semi-supervised graph classification algorithm (Zhu et al., 2003b) to perform the classification given the different similarity graphs mentioned above.

We also compare our algorithm against the following algorithm
SC-MV: We compare SMUG against the spectral clustering algorithm for data with multiple views (Zhou and Burges, 2007). The algorithm tries to classify data when multiple views of the data are available. The multiple views are represented using multiple homogeneous graphs with a common vertex set, V. In each graph, the edge weights represent similarity between the nodes computed using a single feature type. For our experiments, we used the link similarity graph and the content similarity graph as described above as the two views of the same data. Consider the problem of spectral clustering of a single graph. Given a directed graph $G = (V, E, w)$ with

vertex set V; edge set E; and corresponding edge weights w; assume a random walk defined on G with transition probabilities p and stationary distribution π: Let S denote an arbitrary subset of V; and S^c the complement of S: Define the volumes

$$\text{vol } S = \sum_{v \in S} \pi(v) \text{ , and vol } \partial S = \sum_{u \in S, v \in S^c} \pi(u) p(u,v) \tag{3.31}$$

It can be shown that vol ∂S+vol $\partial S^c = 1$, and vol∂S =vol∂S^c. Then a clustering can be obtained by

$$argmin_{\emptyset \neq S \subset V} \frac{\text{vol } \partial S}{\text{vol } \partial S \text{vol } \partial S^c} \tag{3.32}$$

The intuition behind this cut is as follows. Assume a random web surfer who browses web pages by following hyperlinks and occasionally jumping to a randomly chosen web page. Then the web surfer will regard a set of hyperlinked web pages as a community if the probability of leaving the web page set is small while the stationary probability mass of the same subset is large.

This approach is naturally extended to cluster multiple graphs. Assume two directed graphs $G_i = (V, E_i, w_i)$ for $i = 1, 2$ which share the same set of vertices while having different edges and weights. Suppose S to be a nonempty subset of V: Define

$$\text{mvol } S = \alpha \text{vol }_1 S + (1 - \alpha) \text{vol }_2 S \tag{3.33}$$

and

$$\text{mvol } \partial S = \alpha \text{vol }_1 \partial S + (1 - \alpha) \text{vol }_2 \partial S \tag{3.34}$$

ACL id1	**ACL id2**	**Link Similarity Score**
W05-0817	W05-0819	0.775
W05-0812	W05-0819	0.612
H05-1010	H05-1011	0.584
W06-2929	W06-2935	0.577
N07-1064	W07-0732	0.577
W06-2923	W06-2929	0.577
P98-1069	P99-1067	0.561
P91-1022	P91-1023	0.556
W07-0728	W07-0732	0.534
C86-1028	C86-1109	0.516

Table 3.1: Top similar pairs of papers according to link similarity in the AAN data set

where α is a parameter in $[0, 1]$. Then we can cluster the vertex set V into two subsets by

$$argmin_{\emptyset \neq S \subset V} \frac{\text{mvol } \partial S}{\text{mvol } \partial S \text{mvol } \partial S^c} \tag{3.35}$$

Clearly, the case of $\alpha = 0$ or 1 reduces to the cut for a single graph. The basic motivation of defining such a multiple graph cut is that we want to obtain a cut which is good on average while it may not be the best for a single graph. The parameter α is used to specify the relative importance of each graph in clustering. It is not hard to imagine that the relative importance measure varies across different clustering goals. Note that SMUG also has parameters to specify the importance of each graph. In addition to that, SMUG also has parameters to specify the amount of dependency between the different graphs.

3.5.2 Experiment II

We illustrate the improved performance of our similarity measure in the context of clustering. We compare our similarity measure against the three similarity baselines mentioned above. We use a spectral graph clustering algorithm (Dhillon et al., 2007) to perform the clustering. (Dhillon et al., 2007) formulate the clustering problem as a k-way normalized cut problem. Given a graph $G = (V, E, A)$, where V is the set of vertices, E is the set of edges connecting vertices and A is the edge affinity matrix. Therefore $A[i][j]$ corresponds to the edge weight between vertices i and j. Assume A is non-negative and symmetric. Suppose $C_1, C_2 \subset V$, the number of links between the two sets, C_1 and C_2 is computed as,

$$L(C_1, C_2) = \sum_{i \in C_1, j \in C_2} A[i][j] \tag{3.36}$$

The Normalized Link Ratio (NLR) of C_1, C_2 is defined as,

$$NLR(C_1, C_2) = \frac{L(C_1, C_2)}{L(C_1, V)} \tag{3.37}$$

The k-way normalized cut problem is to minimize the links that escape a cluster relative to the total weight of the cluster. For a k-way partitioning of the vertices $(C_1, C_2, \ldots, C_k)$, they solve the following optimization problem

$$\text{minimize } \frac{1}{k} \sum_{i=1}^{k} NLR(C_i, V\ C_i) \tag{3.38}$$

A spectral relaxation to this problem is obtained as follows: let D be the diagonal matrix whose $(i,i)^{th}$ entry is the sum of the entries of row i in matrix A, $D[i][i] = \sum_{j=1}^{n} A[i][j]$. The normalized cut criterion is equivalent to the following problem:

$$\text{maximize } \frac{1}{k} Trace(Z^T A Z) \tag{3.39}$$

where $Z = X(X^T D X)^{-\frac{1}{2}}$, and X is an $n \times k$ indicator matrix for the partition. Note that $Z^T D Z = I_k$.

Letting $\hat{Z} = D^{\frac{1}{2}} Z$ and relaxing the constraint that X is an indicator matrix results in the following problem: maximize the trace of $\hat{Z} D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \hat{Z}$, where the constraints on $\hat{Z}$ are relaxed such that $\hat{Z}^T \hat{Z} = I_k$. A well-known solution to this problem is obtained by setting the matrix $\hat{Z}$ to be the top k eigenvectors of the matrix $D^{\frac{1}{2}} A D^{-\frac{1}{2}}$. These eigenvectors are then used to compute a discrete clustering of the nodes.

We compare the clustering against an unsupervised co-clustering algorithm (Dhillon et al., 2003). Most clustering algorithms focus on one-way clustering, i.e., cluster one dimension of the table based on similarities along the second dimension. For example, documents may be clustered based upon their word distributions or words may be clustered based upon their distribution amongst documents.It is often desirable to co-cluster or simultaneously cluster both dimensions of a document-term occurrence matrix by exploiting the clear duality between rows and columns. For example, we may be interested in nding similar documents and their interplay with word clusters.

They treat the (normalized) non-negative document-term matrix as a joint probability distribution between two discrete random variables that take values over the rows and columns. We dene co-clustering as a pair of maps from rows to row-clusters and from columns to column-clusters. Clearly, these maps induce clustered random variables. Information theory can now be used to give a theoretical formulation to the problem: the optimal co-clustering is one that leads to the largest mutual information between the clustered random variables. Equivalently, the optimal co-clustering is one that minimizes the difference ("loss") in mutual information between the original random variables and the mutual information between the clustered random variables.

They propose an algorithm that intertwines both row and column clustering at all stages. Row clustering is done by assessing closeness of each row distribution, in relative entropy, to certain "row cluster prototypes". Column clustering is done similarly, and this process is iterated till it converges to a local minimum.

Let X and Y be discrete random variables that take values in the sets $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_m\}$ respectively. Let $p(X, Y)$ denote the joint probability distribution between X and Y . We will think of $p(X, Y)$ as an $n \times m$ matrix. They propose the problem of co-clustering as clustering X into k clusters and Y into l clusters. The tuple (C_X, C_Y) is referred to as a co-clustering. Suppose we are given a co-clustering. The rows of the joint distribution p can be re-ordered such that all rows mapping into $\hat{x_1}$ are arranged rst, followed by all rows mapping into $\hat{x_2}$, and so on. Similarly, the columns of the joint distribution p are re-ordered such that all columns mapping into $\hat{y_1}$ are arranged rst, followed by all columns mapping into $\hat{y_2}$, and so on. This row-column reordering has the effect of dividing the distribution p into little two-dimensional blocks. Each such block is referred to as a co-cluster.

Let the clusterings of X and Y be denoted as C_X and C_Y respectively.

$$C_X : \{x_1, x_2, \ldots, x_n\} \rightarrow \{\hat{x_1}, \hat{x_2}, \ldots, \hat{x_k}\} \tag{3.40}$$

and

$$C_Y : \{y_1, y_2, \ldots, y_n\} \rightarrow \{\hat{y_1}, \hat{y_2}, \ldots, \hat{y_l}\} \tag{3.41}$$

where $\hat{x_i}$ and $\hat{y_j}$ refer to a single cluster. A fundamental quantity that measures the amount of information random variable X contains about Y (and vice versa) is the mutual information $I(X;Y)$ (Cover and Thomas, 1991). The quality of a co-clustering is judged by the resulting loss in mutual information, $I(X;Y) - I(\hat{X};\hat{Y})$. Therefore, an optimal co-clustering minimizes $I(X;Y) - I(\hat{X};\hat{Y})$ subject to the constraints on the number of row and column. clusters. They show that the problem of minimizing mutual information is equivalent to minimizing Kullback-Leibler divergence between two distributions as follows

$$I(X;Y)I(\hat{X},\hat{Y}) = D(p(X,Y)||q(X,Y)) \tag{3.42}$$

where $q(X,Y)$ is defined as

$$q(x,y) = p(\hat{x},\hat{y})p(x|\hat{x})p(y|\hat{y}), \text{ where } x \in \hat{x},\ y \in \hat{y} \tag{3.43}$$

They propose an iterative greedy algorithm for the minimization problem. For each of the data sets used in our experiments, we use a word-document co-occurrence matrix as the input to the co-clustering algorithm. We set the number of clusters for the

Data Set	**#Nodes**	**#Edges**	**#Keywords**	**#Classes**
AAN	380	3	572	4221
WebKB	877	1608	1703	5
Cora	2708	5429	1433	7

Table 3.2: Statistics of different data sets

documents to the number of classes in the data set. However, it is not clear how to set the number of word clusters. Therefore, we try $l = 2, 4, 8, 16$ word clusters. We represent the algorithm which uses i word clusters as CoC$_i$.

For all experiments, we set $\alpha_0 = 0.4$. This is because the initial similarity estimates are very coarse and computed using naive methods like cosine similarity. The similarity measures in all feature spaces are sparse. For example, there are $72,010$ pairs of publications in the AAN data set, of which only $15,534$ pairs of publications have a non-zero similarity using content similarity. There are $10,233$ pairs of publications which have a non-zero similarity value when using citation similarity. Therefore, we wanted to learn more similar pairs using the dependency between the different feature space which leads to a lower value of α_0.

We conducted all our experiments on the following data sets.

- AAN Data: The ACL Anthology(Bird et al., 2008) is a collection of papers from the Computational Linguistics journal as well as proceedings from ACL conferences and workshops and includes $17,610$ papers. To build the **A**CL **A**nthology **N**etwork (AAN), (Radev et al., 2009b) manually performed some preprocessing tasks including parsing references and building the network metadata, the citation, and the author collaboration networks. A complete description of the

Keyphrase	**Keyphrase**	**Similarity Score**
alignment error rate	error rate aer	0.864
source language	target language	0.792
source side	target side	0.720
statistical machine translation	translation model	0.730
statistical machine translation	smt models	0.745
statistical machine translation	word alignment	0.690
alignment models	word alignment	0.688
bleu score	statistical machine translation	0.669
alignment models	statistical machine translation	0.657
machine learning	training data	0.644
phrase-based translation	translation model	0.641
dependency parser	dependency parsing	0.621
dependency parsing	dependency trees	0.612
statistical parsers	statistical parsing	0.609
penn treebank	statistical parsing	0.594
dependency parser	head word	0.590
headline generation	summarization system	0.572
document compressions	important sentences	0.570
discourse structure	dependency parsing	0.558
rouge score	multidocument summarization	0.540

Table 3.3: Top similar pairs of keyphrases extracted from the publications in the AAN data set

AAN dataset can be found in Appendix B

We chose a subset of papers in 3 topics (Machine Translation, Dependency Parsing, Summarization) from the ACL anthology. These topics are three main research areas in Natural Language Processing (NLP). Specifically, we collected all papers which were cited by papers whose titles contain any of the following phrases, "'Dependency Parsing"', "'Machine Translation"', "'Summarization"'. From this list, we removed all the papers which contained any of the above phrases in their title because this would make the clustering task easy. The pruned list contains 1190 papers. We manually classified each paper into four classes (Dependency Parsing, Machine Translation, Summarization, Other) by considering the full text of the paper. The manually cleaned data set consists of 275 Machine Translation papers, 73 Dependency Parsing papers and 32 Summarization papers. Table 3.5 lists a few sample papers from each class.

For each publication, we created a feature vector description in order to compute content similarity. We removed the reference and acknowledgments section from each publication's text as a preprocessing step. After removing stopwords, all bigrams and trigrams with document frequency less than 10 were removed to create a dictionary with 1072 n-grams. Each publication in the dataset is described by the top 30 most frequent bigrams and trigrams in the text of the publication. Note that only bigrams and trigrams which are present in the dictionary are used for representing each publication. Table 3.4 shows the extracted features from an example publication from each class. Table 3.3 shows the top similar pairs of keyphrases in the dictionary.

J03-3002 (**Machine Translation**)	**C04-1159** (**Dependency Parsing**)	**P02-1057** (**Summarization**)
english arabic	sentence boundaries	perfect discourse
candidate pairs	dependency structure	discourse trees
english chinese	structure analysis	source model
translation lexicon	sentence boundary	summary quality
content based	boundary detection	mitre data
candidate pair	dependency information	shaky ground
parallel corpora	boundary detection	discourse units
same meaning	structure analysis	possible compressions
web pages	dependency relationships	syntactic constituents
language pair	japanese speech	longer documents
competitive linking	dependency structure analysis	channel model
parallel corpus	sentence boundary detection	discourse structure
word pairs	target word	important information
word level translation	discourse relationship	original text
language pairs	long distance	maximum likelihood
french english	language models	posterior probability
structural features	machine learning	document compression
statistical mt package	intra sentential dependency	document summarization
parallel data	intra sentential	single document summary
web based parallel	crossed dependencies	extraction based summarizers
candidate document pairs	beam search	important sentence
decision tree	punctuation marks	Headline based
english side	surface expressions	sentence simplification
cross language ir	pause duration	unimportant information
equiprobability assumption	sentence boundary candidates	hierarchical models
cross lingual	text chunking	syntactic structure
human translated	using dependency information	compression system
parallel web	long distance dependencies	statistical hierarchical model
dynamic programming	japanese speech	wall street
sentence pairs	dependency probability	statistically significant

Table 3.4: Example phrases extracted from a publication from each class in the AAN dataset

ACL-ID	Paper Title	Research Topic
W05-0812	Improved HMM Alignment Models for Languages With Scarce Resources	Machine Translation
P07-1111	A Re-Examination of Machine Learning Approaches for Sentence-Level MT Evaluation	Machine Translation
P03-1054	Accurate Unlexicalized Parsing	Dependency Parsing
P07-1050	K-Best Spanning Tree Parsing	Dependency Parsing
P88-1020	Planning Coherent Multi-Sentential Text	Summarization

Table 3.5: Details of a few sample papers classified according to research topic

- **WebKB**(Sen et al., 2008): The data set consists of a subset of the original WebKB data set. The corpus consists of 877 web pages collected from four different universities. Each web page is represented by a 0/1-valued word vector with 1703 unique words after stemming and removing stopwords. All words with document frequency less than 10 were removed.

- **Cora**(Sen et al., 2008): The Cora dataset consists of 2708 scientific publications in the field of machine learning classified into one of the following seven classes: Case Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning, Theory. The papers were selected in a way such that every paper cites or is cited by atleast one other paper. The citation network consists of 5429 links. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. After stemming and removing stopwords, all words with document frequency less than 10 were removed. The dictionary consists of 1433 unique words.

Machine Translation	**Dependency Parsing**	**Summarization**
machine translation	natural language	rhetorical structure
statistical machine translation	wall street journal	novel approach
machine translation system	penn treebank	system for generating
translation quality	dependency structures	summarization system
word alignment	dependency structure analysis	source document
bleu score	kyoto university corpus	newspaper articles
parallel texts	dependency tree	multidocument summarization
experimental results	shared task	headline generation system
parallel corpora	dependency grammar	important sentences
language pairs	parsing accuracy	proposed method
target Language	training data	discourse structure
ibm model	lexical items	document compressions

Table 3.6: Top few words from each cluster extracted from the keyword similarity graph

Table 3.2 summarizes the statistics of the different data sets used in experiments. For each of the data sets, we constructed two graphs, a kewyord feature similarity graph and a link similarity graph. The keyword feature layer graph, $G_f = (V_f, E_f, w_f)$ is a weighted graph where V_f is the set of all features. The edge weight between the features f_i and f_j represents the similarity between the features. The edge weight between two keywords are estimated using the co-occurrence data between the keywords in the data set. Let keyphrase k_i occur n_i times and co-occur with keyphrase k_j, c_ij times, then the similarity between keyphrases, k_i and k_j is computed using the generalized similarity index (van Eck and Waltman, 2009) as follows

$$sim(k_i, k_j) = 2^{\frac{1}{p}} \frac{c_{ij}}{(n_i^p + n_j^p)^{\frac{1}{p}}} \tag{3.44}$$

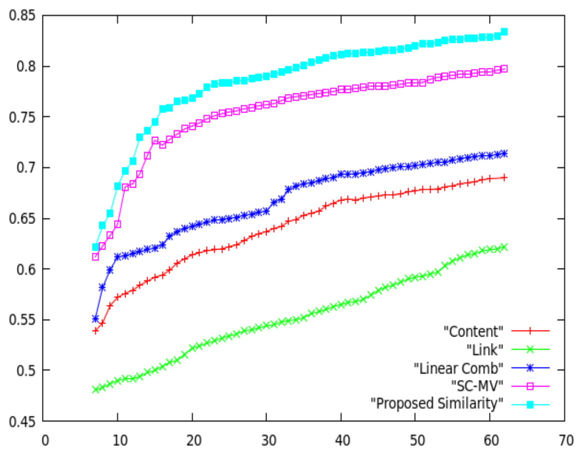
We set $p = 0$, which leads to

$$\lim_{p \to 0} sim(k_i, k_j) = \frac{c_{ij}}{\sqrt{n_i n_j}} \tag{3.45}$$

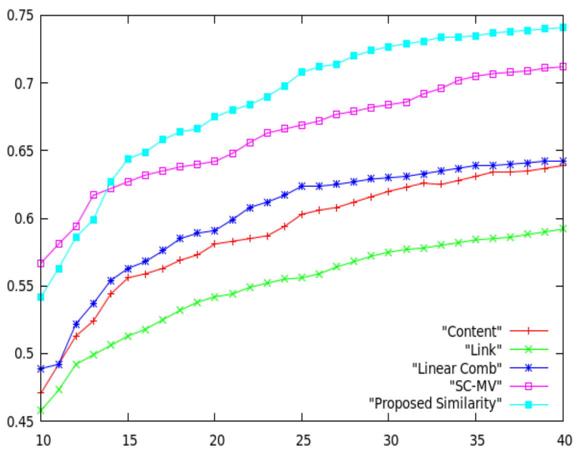
The link similarity graph, $G_o = (V_o, E_o, w_o)$ is a weighted graph where V_o is the set of objects. The edge weight represents the similarity between the objects and is initialized to the link-based similarity. The link similarity between two objects is computed using the similarity measure proposed by (Harel and Koren, 2001) on the object link graph.

We evaluate SMUG in terms of classification accuracy for the classification task. Classification accuracy is the percentage of objects which are correctly classified. Let the true label of a document d_i be denoted by $Y(d_i)$ and the label outputted by the classification algorithm be represented as $\hat{(Y)}(d_i)$. A document is correctly classified if $Y(d_i) = \hat{(Y)}(d_i)$. Let t be the number of documents that are correctly classified and N be the total number of documents to be classified. The classification accuracy $CA = \frac{t}{N}$.

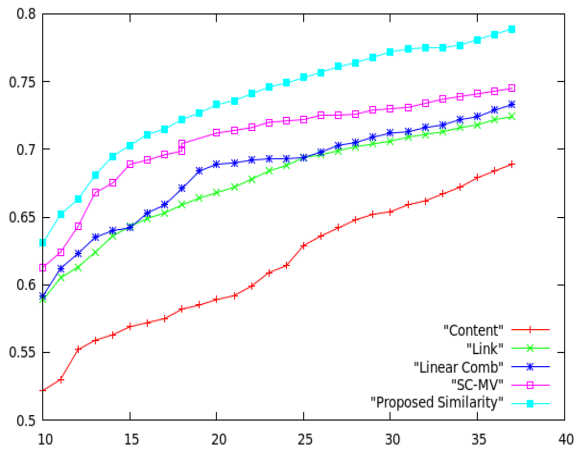
For the clustering task, we use Normalized Mutual Information (Strehl and Ghosh, 2002) as the measure of clustering accuracy. Mutual Information is a symmetric measure which quantifies the statistical information between two distributions. Let $I(X, Y)$ denote the amount of mutual information between the distributions X and Y. Since I(X,Y) has a lower bound of zero and no upper bound, we normalize the quantity using the entropy of X and Y. Thus the normalized mutual information used



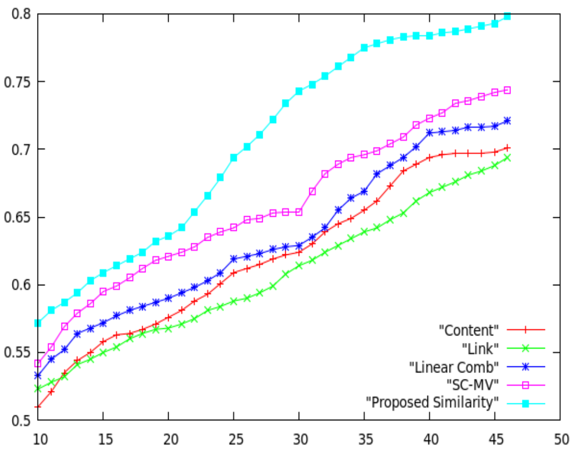
(a) AAN



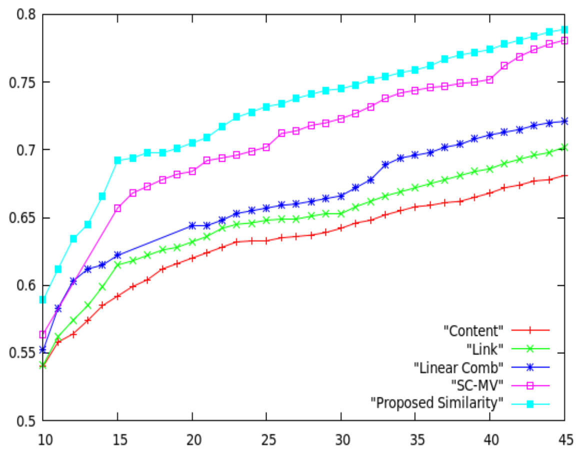
(b) Cornell



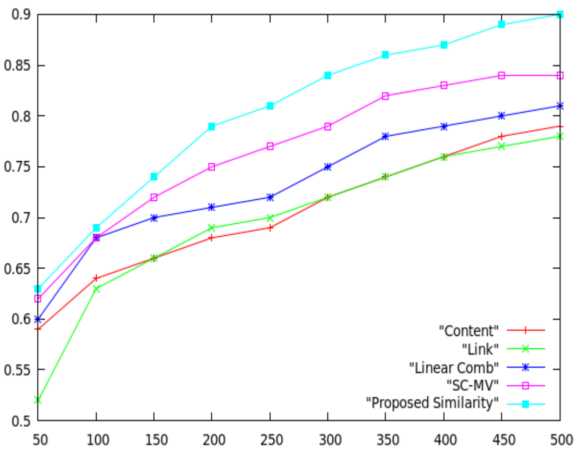
(c) Texas



(d) Washington



(e) Wisconsin



(f) Cora

Figure 3.2: Classification Accuracy on the different data sets. The number of points labeled is plotted along the x-axis and the y-axis shows the classification accuracy on the unlabeled data.

is,

$$NMI(X,Y) = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}} \tag{3.46}$$

We estimate NMI using the samples provided by the clusterings. Let n be the total number of objects (nodes) to be clustered. Let $k(A)$ be the number of clusters according to clustering A and let $k(B)$ be the number of clusters according to clustering B. Let n_h^A denote the number of objects in cluster C_h according to clustering A , and let n_l^B denote the number of objects in cluster C_l according to clustering B . Let $n_{h,l}$ denote the number of objects that are in cluster C_h according to algorithm A as well as in group C_l according to clustering B. The NMI is estimated according to 3.46

$$\widehat{NMI(A,B)} = \frac{\sum_{h=1}^{k(A)} \sum_{l=1}^{k(B)} n_{h,l} log(\frac{n.n_{h,l}}{n_h^A n_l^B})}{\sqrt{\sum_{h=1}^{k(A)} \frac{n_h^A}{n} \sum_{h=1}^{k(B)} \frac{n_h^B}{n}}} \tag{3.47}$$

This evaluation measure has some nice properties such as $NMI(X,X) = 1$. Also $\widehat{NMI(X,Y)} = 1$ if the clusterings X and Y are the same except for a permutation of the cluster labels. $NMI(X,Y) = 0$ if the clustering X is random with respect to Y or vice versa.

Therefore, in our setting, we compare the clusterings obtained by our algorithm X against the clustering Y, induced by the correct cluster labels using NMI. The higher the value the better the clustering obtained. Thus, we compare SMUG against the two above mentioned baselines using NMI as the evaluation metric.

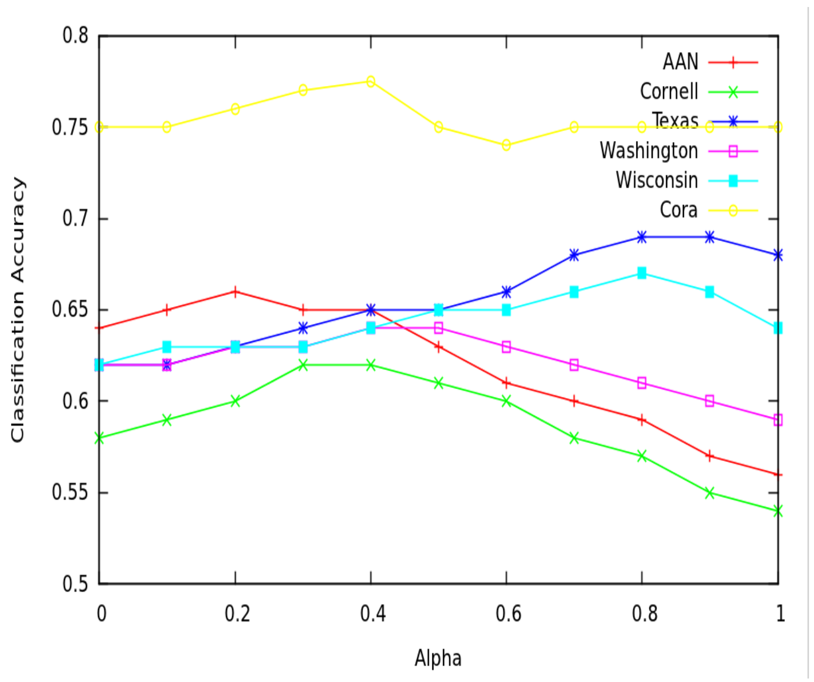


Figure 3.3: Classification accuracy of linear combination of textual similarity and link similarity measures on the different data sets.

Similarity Measure	**AAN**	**Texas**	**Wisconsin**	**Washington**	**Cornell**	**Cora**
Content Similarity (Cosine)	0.66	0.34	0.42	0.59	0.63	0.48
Link Similarity	0.45	0.49	0.39	0.52	0.56	0.52
Linear Combination	0.69	0.54	0.46	0.54	0.68	0.54
CoC_2	0.53	0.42	0.42	0.45	0.46	0.41
CoC_4	0.59	0.46	0.45	0.50	0.48	0.43
CoC_8	0.63	0.48	0.50	0.54	0.52	0.47
CoC_16	0.69	0.55	0.52	0.60	0.56	0.52
SMUG	**0.78**	**0.69**	**0.54**	**0.66**	**0.72**	**0.64**

Table 3.7: Normalized Mutual Information scores of the different similarity measures on the different data sets

3.6 Results

Figure 3.2 shows the accuracy of the classification obtained using different similarity measures. Figure 3.3 shows the accuracy obtained by the linear combination of similarity measures parameterized by α when 10% of the data was used as training data for the classifier. When $\alpha = 0$, the linear combination of similarity measures reduces to link similarity and at $\alpha = 0$ the linear combination of similarity measures is equal to the content similarity. In all the data sets, the link similarity achieves higher accuracy than content similarity, except the Texas data set and the Wisconsin data set. This is primarily due to the vocabulary mismatch in documents from the same class. However, in the Texas and the Wisconsin data sets, there are relative fewer links compared to other data sets and hence the link similarity graph is sparse.

It can be seen that SMUG outperforms all other baselines by a large margin. Note that SMUG and the spectral clustering algorithm on multiple graphs is given the same information and SMUG achieves better performance. We attribute this to the rich representation of the data. In our algorithm, the data is represented as a set of heterogeneous graphs (layers) which are connected together. Thus, we were able to iteratively improve our similarity estimates using similarity estimates from other feature spaces. Whereas, in the case of the algorithm in (Zhou and Burges, 2007) all the graphs are isolated homogeneous graphs. Hence there is no information transfer across the different graphs.

Table 3.7 shows the NMI scores obtained by the different similarity measures on the different data sets.

We also clustered the keyword feature layer of the AAN data set to show that SMUG improves the similarity estimates in all the input graphs. Table 3.6 shows the most frequent keywords from each cluster. It can be seen that the keyword clusters are very cohesive and are indicative of the research area they belong to. This shows that similarity estimates in the keyword feature layer are also improved using similarity estimates from the citation similarity space.

An important property of this framework is that we enhance any coarse-grained initial similarity measure. For example, similarity between keywords can be estimated using more sophisticated methods which use external knowledge bases like WordNet, Wikipedia (Danushka Bollegala, 2007; Strube and Ponzetto, 2006). Although, there exist lots of similarity measures for computing similarity between keywords and phrases, there are relatively few approaches to compute similarity between entire documents using pairwise similarities between keywords (Mihalcea and Corley, 2006). Note that using the pairwise similarities between individual keywords alone, we can estimate the similarity between entire documents using SMUG.

The work is also quite similar to *co-training* (Blum and Mitchell, 1998) in the sense of using multiple views (feature types). The cotraining framework is applicable for any learning problem which uses labeled data. (Blum and Mitchell, 1998) formalize the co-training setting and provide theoretical learning guarantees subject to certain assumptions.

Let D be the distribution over the set of objects X. Each object can be represented by two sets of features, X_1 and X_2. The object space can be represented as $X = X_1 \times X_2$. Each object is represented as $x = (x_1, x_2)$. Consider the problem of classification using the sets of features. Let C_1 and C_2 define the set of target (classification) functions over X_1 and X_2 respectively: $f_1 \in C_1$ and $f_2 \in C_2$. There are two main assumptions made by the cotraining framework.

Definition III.2. The instance distribution D is compatible with the target function $f = (f_1, f_2)$ if for any $x = (x_1, x_2)$ with non-zero probability, $f(x) = f_1(x_1) = f_2(x_2)$.

The compatibility assumption states that each set of features independently is sufficient for learning a perfect classifier. For most examples, the target functions over each feature set predict the same label. For example, in the web page domain, the class of the instance should be identifiable using either the link structure or the page text alone. The second assumption is that the features in one set of an instance are conditionally independent of the features in the second set, given the class of the instance. Mathematically, a pair of views (x_1, x_2) satisfy view independence if,

$$Pr[X_1 = x_1 | X_2 = x_2, Y = y] = Pr[X_1 = x_1 | Y = y] \tag{3.48}$$

```
Input: a set $L$ of labeled training examples
a set U of unlabeled examples
Create a pool U' of examples by choosing u examples at random from U.
**while** loop for k iterations **do**
   Use $L$ to train a classifier $h_1$ that considers only $X_1$
   Use $L$ to train a classifier $h_2$ that considers only $X_2$
   Allow $h_1$ to label $p$ positive and $n$ negative examples from U'
   Allow $h_2$ to label $p$ positive and $n$ negative examples from U'
   Add these self-labeled examples to $L$
   Randomly choose $2p + 2n$ examples from U to replenish U'
**end while**
```

Figure 3.4: Cotraining algorithm (Blum and Mitchell, 1998)

and

$$Pr[X_2 = x_2 | X_1 = x_1, Y = y] = Pr[X_2 = x_2 | Y = y] \tag{3.49}$$

A classification problem instance is said to satisfy view independence if all pairs (x_1, x_2) satisfy view independence. This assumes that the words in a publication are not related to the links, except through the class of the publication, an unrealistic assumption in practiceNigam and Ghani (2000); Balcan and Blum (2005). The co-training algorithm is shown in Figure 3.4

The cotraining algorithm learns a classifier by comining classifiers learnt using individual feature spaces using unlabeled data. Intuitively, the classifier h_1 adds examples to the labeled set that h_2 will be able to use for learning and vice-versa. Similarly, in the SMUG framework, we use the information about which pair of features are similar to learn object similarity.

However, there are a few fundamental differences between the SMUG framework and the cotraining framework. SMUG is motivated by applications where the main assumptions of the cotraining framework do not hold. In most of the classification data sets used in our experiments, the compatibility assumption does not hold, i.e, no single feature can be used to build a perfect classifier. This is due to the sparsity of similarity values in any feature space. For example, many pairs of publications from the same class do not have a non-zero similarity value because of vocabulary mismatch.

We also rely on the *dependence* between the feature spaces in order to learn similarity using the different existing similarity measures. For example, we use the information that two publications that are similar because of link similarity to increment the similarity value between the publications' corresponding features.

For example, when we apply the cotraining algorithm for classifying publications using keywords and citations as the two feature spaces, we learn which publications belong to a particular class in the citation feature space using labeled examples generated by the classifier trained using keyword features. However, the SMUG framework uses feature similarity to learn object similarity. This is analogous to the comparison between feature labeling vs instance labeling in the active learning framework. (Raghavan et al., 2006; Druck et al., 2009) show that intelligently soliciting labels on multiple features facilitates more efficient annotation and leads to higher classification accuracy. This shows that learning at the feature level can lead to better or at least complement learning at the object level.

CHAPTER IV

Simultaneously Optimizing Feature Weights and Similarity

4.1 Introduction

There has been a lot of work on learning feature weights as part of similarity metric learning (Schultz and Joachims, 2003; Davis et al., 2007b). To the best of our knowledge, there exists no work which uses multiple sources of similarity to better learn feature weights and use the same to reinforce similarity estimates in the different feature spaces.

4.2 Motivation

First, we explain how similarity learning and feature weight learning can mutually benefit from each other using an example. For example, consider the following two publications in the field of Machine Translation

- Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A Statistical Approach to Machine Translation. Computational Linguistics. Volume 16. Number 2. 1990.

- William A. Gale and Kenneth Ward Church. A Program For Aligning Sentences In Bilingual Corpora. In Proceedings of ACL. 1991.

- Daniel Marcu and William Wong. A Phrase-Based, Joint Probability Model for Statistical Machine Translation. In Proceedings of EMNLP 2002.

Clearly, all the papers belong to the field of *Machine Translation* but the second paper contains the phrase "Machine Translation" only once in the entire text. However, we can learn to attribute some similarity between the first publication and the second publication using the third publication's data. The keywords "Bilingual Corpora" and "Machine Translation" co-occur in the third publication's text which makes the keywords themselves similar. We attribute some similarity between the first and second publication since they contain similar keywords. Of course, the attributed similarity is proportional to the number of co-occurrences. This shows how similarity learning can benefit from good keywords.

Now, assume that "Machine Translation" is an *important* keyword (high feature weight) for the first and third publication while the keyword "Bilingual Corpora" has a low feature weight for the second publication. We explained how to infer similarity between the first and second publication using the third publication as a bridge. Using the newly learned similarity measure, we can infer that "Bilingual Corpora" is an *important* keyword for the second publication since a similar keyword ("Machine Translation") is an important keyword for similar publications.

Let documents D_i and D_j contain keywords K_{ik} and K_{jl}. Then intuitively, the similarity between two documents should be jointly proportional to

- The similarity between keywords K_{ik} and K_{jl}
- The importance of K_{ik} to D_i and importance of K_{jl} to D_j.

Similarly the importance of candidate keyword K_{ik} to document D_i should be jointly proportional to

- The similarity between documents D_i and D_j.
- The similarity between keyphrases K_{ik} and K_{jl} and importance of K_{jl} to D_j.

In the next two sections, we formalize and exploit this observation to simultaneously optimize similarity between documents and feature weights of keywords in a principled way.

4.3 Problem Formulation

We assume that a set of keywords have been extracted for the set of documents to be analyzed. Documents are represented by a set of keywords. In addition to that, we have crude "initial" similarities estimated between documents and also between keywords and an "importance score" (interchangeably used with feature weight) of a keyword to the document. The similarities and importance scores are represented using two layers of graphs. We formally define the necessary concepts,

Definition 1: Documents and corresponding keywords

We have a set of N documents $D = \{d_1, d_2, \ldots, d_N\}$. Each document, d_i has a set of m_i keywords $K_i = \{k_{i1}, k_{i2}, \ldots, k_{im_i}\}$

Definition 2: Document Similarity Graph

The document similarity graph, $G_1 = (V_1, E_1)$, consists of the set of documents as nodes and the edge weights represent the initial similarity between the documents.

Definition 3: Keyword Similarity Graph

The keyword similarity graph, $G_2 = (V_2, E_2)$, consists of the set of keywords as nodes and the edge weights represent the initial similarity between the keywords.

The document similarity graph and the keyword similarity graph can be considered as two layers of graphs which are connected by the function defined below

Definition 4: Keyword Importance Scores (KIS)

There exists an edge between d_i and k_{ij} for $1 \leq j \leq m_i$. Let Z represent the keyword importance function, i.e, $Z_{d_i,k_{ij}}$ represents the importance score for keyword k_{ij} to document d_i.

4.4 Regularization Framework

The framework for mutual learning of similarity and feature weights is an extension of the framework described in Chapter III. In this framework, objects with multiple heterogeneous features are represented using multiple layers of graphs. One layer is used to represent the objects and other layers are used to represent the different feature types. The edge weights in the same layer are used to represent similarity scores between objects or features. An edge exists between two features if they co-occur in any particular object's feature vector. An edge exists between an object and a feature if the feature exists in the object's description. However, the edges across layers are unweighted and the objective function in Chapter III is defined only over the edge weights in the same layer. Hence, the framework does not fully exploit the available information and is not robust to noisy features, since all the features are equally weighted. A keyword which is not related to the semantics of the paper contributes equally to the similarity between two publications as much as a relevant keyword. For example, the contribution of the keyword "Related Work" should be minimal to the similarity between any two publications since it is often not relevant to the publication's research area. In order to learn feature weights, we define an objective function over the edge weights in each layer and the *KIS* as follows

$$\begin{aligned}\Omega(w,Z) &= \alpha_0 * ISC(w,w*) + \alpha_1 * IKC(Z,Z*) \\ &\quad + \alpha_2 * KS(w,Z) + \alpha_3 * SK(Z,w) \end{aligned} \tag{4.1}$$

where $\alpha_0 + \alpha_1 + \alpha_2 + \alpha_3 = 1$.

ISC refers to **I**nitial **S**imilarity **C**constraint and *IKC* refers to **I**nitial **K**eyphrase importance **C**onstraint. They are defined as follows

$$ISC(w, w*) = \sum_{u,v \in G_1} (w_{u,v} - w^*_{u,v})^2 \tag{4.2}$$

$$IKC(Z, Z*) = \sum_{u \in G_1, v \in G_2} (Z_{u,v} - Z^*_{u,v})^2 \tag{4.3}$$

KS refers to **K**eyphrase importance induced **S**imilarity and *SK* refers to **S**imilarity induced **K**eyphrase importance. They are defined as follows

$$KS(w, Z) = \sum_{u_1,v_1 \in G_1} \sum_{u_2,v_2 \in G_2} Z_{u_1,u_2} Z_{v_1,v_2} (w_{u_1,v_1} - w_{u_2,v_2})^2 \tag{4.4}$$

and

$$SK(w, Z) = \sum_{u_1,v_1 \in G_1} \sum_{u_2,v_2 \in G_2} w_{u_1,v_1} w_{u_2,v_2} (Z_{u_1,u_2} - Z_{v_1,v_2})^2 \tag{4.5}$$

The problem is to minimize the objective function defined in equation 4.1. The objective function consists of four parts. The first and second parts are initial similarity constraint and initial keyword constraint. They ensure that the optimized edge weights are close to the initial edge weights. The weights α_0 and α_1 ensure that the optimized weights are close to the initial weights, in other words, they represent the confidence level in initial weights.

The significance of the third and the fourth parts of the objective function are best explained by a simple example. Consider two graphs, G_1 and G_2. Let G_1 be the graph containing publications as nodes and edge weights representing initial similarity values. Let G_2 be the graph corresponding to keywords and edge weights represent similarity between keywords. There is an edge from a node u_1 in G_1 to a node v_1 in G_2 if the publication corresponding to u_1 contains the keyword corresponding to v_1.

Minimizing the keyword importance induced similarity part corresponds to updating similarity values between keywords in proportion to importance of the keywords to the respective documents they are contained in and the similarity between the documents. keyword importance induced similarity part also helps updating similarity values between documents in proportion to importance of keywords they contain and the similarity between the contained keywords.

Minimizing the similarity induced keyword part corresponds to updating importance scores of keywords to documents in proportion to the following

- Similarity between v_1 and other keywords $v_2 \in G_2$
- Importance values of v_2 to documents $u_2 \in G_1$
- Similarity between u_1 and u_2

Therefore, even if the frequency of a keyword such as “Machine Translation” in a publication is not high, it can achieve a high importance score if it contains many other similar keywords such as “Bilingual Corpora” and “Word alignment” and is similar to other publications which contain the keyword “Machine Translation” in another feature space.

4.5 Efficient Algorithm

We seek to minimize the objective function using Alternating Optimization (AO) (Bezdek and Hathaway, 2002), an approximate optimization method. Alternating optimization is an iterative procedure for minimizing (or maximizing) the function $f(x) = f(X_1, X_2, \ldots, X_t)$ jointly over all variables by alternating restricted minimizations over the individual subsets of variables $X_1, \ldots, X_t$.

In this optimization method, we partition the set of variables into a set of mutually exclusive, exhaustive subsets. We iteratively perform minimizations over each subset of variables while maintaining the other subsets of variables fixed. Formally, let the set of real-valued variables be $X = \{X_1, X_2, \ldots, X_N\}$ be partitioned into m subsets, $\{Y_1, Y_2, \ldots, Y_m\}$. Let $s_i = |Y_i|$. Then we begin with the initial set of values $\{Y_1^0, {Y_2}^0, \ldots, {Y_m}^0\}$ and make restricted minimizations of the following form,

$$\min_{Y_i \in \mathbb{R}^{s_i}} \{f(\underline{Y_1}^{r+1}, \ldots, \underline{Y_{i-1}}^{r+1}, Y_i, \underline{Y_{i+1}}^{r}, \ldots, \underline{Y_m}^{r})\} \tag{4.6}$$

where $i = 1, 2, \ldots, m$. The ***underline notation*** $\underline{Y_j}$ indicates that the subset of variables Y_j are fixed with respect to Y_i. In the context of our problem, we update each edge weight while maintaining other edge weights to be a constant. Then the problem boils down to the minimization problem over a single edge weight. For example, let us solve the minimization problem for edge weight corresponding to (u_i, v_j) where $u_i, v_j \in G_1$ (The case where $u_i, v_j \in G_2$ is analogous). Clearly the objective function is a convex function in w_{u_i,v_j}. The partial derivative of the objective function with respect to the edge weight is given below,

$$\begin{aligned}\frac{\partial\Omega(w,Z)}{\partial w_{u_i,v_j}} &= 2\alpha_0(w_{u_i,v_j} - w^*_{u_i,v_j})\\ &+2\alpha_2 * \sum_{u_2,v_2\in G_2}(w_{u_i,v_j} - \underline{w_{u_2,v_2}})\underline{Z_{u_1,u_2}}\underline{Z_{v_j,v_2}}\\ &+\alpha_3 * \sum_{u_2,v_2\in G_2}(\underline{Z_{u_i,u_2}} - \underline{Z_{v_j,v_2}})^2 w_{u_i,v_j}\underline{w_{u_2,v_2}}. \qquad (4.7)\end{aligned}$$

To minimize the above function, we set the partial derivative to zero which gives us the following expression,

$$w_{u_j,v_k} = \frac{1}{C_1}(\alpha_0 w^*_{u_i,v_j} + \alpha_2 \sum_{u_2,v_2\in G_2} \underline{Z_{u_i,u_2}}\ \underline{w_{u_2,v_2}}\ \underline{Z_{v_j,v_2}}) \qquad (4.8)$$

where

$$\begin{aligned}C_1 = \quad & \alpha_0 + \alpha_2 \sum_{u_2,v_2\in G_2} \underline{Z_{u_i,u_2}}\ \underline{Z_{v_j,v_2}}\\ & +\frac{\alpha_3}{2}\sum_{u_2,v_2\in G_2}(\underline{Z_{u_i,u_2}} - \underline{Z_{v_j,v_2}})^2\underline{w_{u_2,v_2}}\end{aligned}$$

Similarly, we can derive the update equation for importance scores, Z_{u_i,u_j} as below

$$\begin{aligned}Z_{u_i,u_j} \quad &= \frac{1}{C_2}(\alpha_1 Z^*_{u_i,u_j} +\\ & \alpha_3 \sum_{v_1\in G_1}\sum_{v_2\in G_2} \underline{w_{u_i,v_1}}\ \underline{w_{u_j,v_2}}\ \underline{Z_{v_1,v_2}}) \qquad (4.9)\end{aligned}$$

where

$$\begin{aligned} C_2 = \quad & \alpha_1 + \alpha_3 \sum_{v_1 \in G_1} \sum_{v_2 \in G_2} \underline{w_{u_i,v_1}}\ \underline{w_{u_j,v_2}} \\ & + \frac{\alpha_2}{2} \sum_{v_1 \in G_1} \sum_{v_2 \in G_2} (\underline{w_{u_i,v_1}} - \underline{w_{u_j,v_2}})^2 \underline{Z_{v_1,v_2}} \end{aligned}$$

The similarity score between two nodes is proportional to the similarity between nodes in the other layer. For example, the similarity between two documents and keywords is proportional to the similarity between the keywords and the documents they are contained in respectively. C_1 and C_2 play the role of a normalization constant. Therefore, for similarity between two nodes to be high, it is required that they not only contain a lot of similar nodes in the other graph but the similar nodes need to be important to the two target nodes.

Similarly, a particular keyword is important to a document if similar keywords are important to similar documents. It is necessary that the similarity between the keywords and the documents is high.

It can be shown that equations 4.8 and 4.9 converge q-linearly since the minimization problem is convex in each of the variables individually and hence has a global and unique minimizer (Bezdek and Hathaway, 2002).

4.5.1 Layered Random Walk Interpretation

The above algorithm has a very nice intuitive interpretation in terms of random walks over the two different graphs. Assume the initial weights are transition probability values after the graphs are normalized so that each row of the adjacency matrices sums to 1. Then the similarity between two nodes u and v in the same graph is computed as the sum of two terms. The first term is α_0 times the initial similarity. This is necessary so that the optimized similarity values are not too far away from the initial similarity values. The second term corresponds to the probability of a random walk of length 3 starting at u and reaching v through two intermediate nodes from the other graph.

The semantics of the random walk depends on whether u, v are documents or keywords. For example, if the two nodes are documents, then the similarity between two documents d_1 and d_2 is the probability of a random walk starting at document d_1 and then moving to a keyword k_1 and then moving to keyword k_2 and then to document d_2. Note that keywords k_1 and k_2 can be the same keyword which accounts for similarity between documents because they contain the same keyword.

Note that second and higher order dependencies are also taken into account by this algorithm. That is, two papers may become similar because they contain two keywords which are connected by a path in the keyword graph, whose length is greater than 1. This is due to the iterative nature of the algorithm. For example, the keywords “Machine Translation” and “Bilingual corpora” occur often together and hence any co-occurrence based similarity measure will assign that pair a high initial similarity value. Hence two publications which contain these words will be assigned a non-zero similarity value after a single iteration. “Bilingual corpora” and “SMT” (abbreviation for Statistical Machine Translation) can have a high initial similarity value which enables assigning a high similarity value between “Machine Translation” and “SMT”. This leads to a chain effect as the number of iterations increases which helps assign non-zero similarity values between semantically similar documents even if they do not contain common keywords.

4.5.2 Relation to Node Label Regularization

In this section, we explore connections to node label regularization (Zhu et al., 2003b). Specifically, we would like to know if there exists a transformation to the input graph which would make node label regularization and edge-weight regularization equivalent. Node label regularization is a technique proposed in (Zhu et al., 2003b) for semi-supervised graph classification. The set of labeled and unlabeled nodes in a graph $G = (V, E)$ is denoted as L and U respectively. The label of node i is denoted as $y(i)$. In order to ensure that similar points have similar labels, an energy function is defined over the node labels as follows

$$E(y) = \frac{1}{2} \sum_{i,j} w_{ij} (y(i) - y(j))^2 \tag{4.10}$$

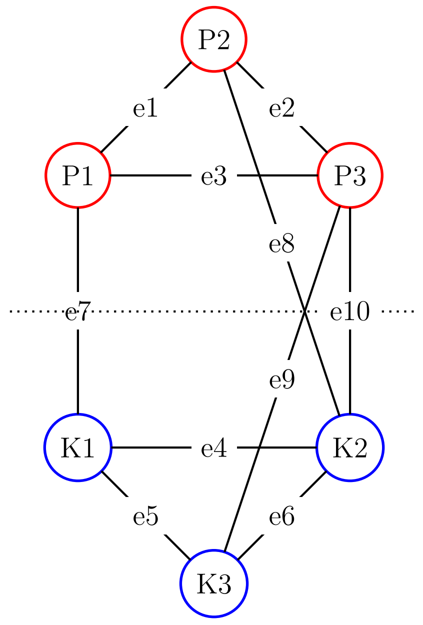
A low energy corresponds to a slowly varying function over the graph. One of the key differences between our technique and the technique in (Zhu et al., 2003b) lies in the values being regularized: edge weights or node labels. We apply a transformation over the graph such that in the transformed graph, the nodes correspond to edges in the original graph. Specifically, we create a new graph $G' = (V', E')$ such that there exists a node $u \in V'$ for every edge $(u_1, v_1) \in E$ in the original graph. There exists an edge $(u, v) \in E'$ if and only if the edges corresponding to u and v share a common node. Figure 4.1 shows a sample graph and the transformed graph.The nodes corresponding to intra-layer edges are labeled using the corresponding node labels in the original graph while nodes corresponding to inter-layer edges in the original graph are labeled as L_i.

Figure 4.1 shows that for the purposes of similarity learning, in the context of node regularization we would like to minimize the squared difference between u's value and v's value. However, they are not neighbors in the transformed graph. Therefore, at the outset it looks like node regularization and edge-weight regularization are fundamentally different. In Chapter III, the inter-layer edge weights or feature weights are maintained constant and the optimization is strictly defined over the intra-layer edge weights. We explore the relation to node label regularization in the following cases.

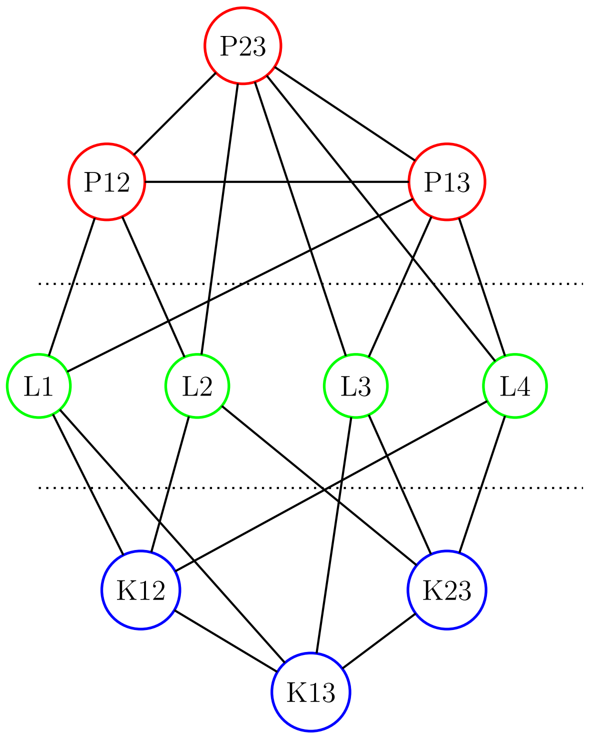
- Case 1: Assume the optimization is strictly defined over the edge weights in each layer or the similarity weights and not the edge weights across layers or feature weights. Let the label of a node corresponding to an intra-layer edge, for example, $P_{1,2}$, be denoted as $y_{P_{1,2}}$ while the weight of nodes corresponding to inter-layer edges, for example, L_1, as w_{L_1} Then the objective function for the sample graph can be rewritten in terms of node labels.

$$\begin{aligned}\Omega(y) =& (y_{P_{12}} - y_{K_{12}})^2(w_{L1}w_{L2}) + (y_{P_{13}} - y_{K_{12}})^2(w_{L1}w_{L4}) \\ & (y_{P_{13}} - y_{K_{13}})^2(w_{L1}w_{L3}) + (y_{P_{23}} - y_{K_{23}})^2(w_{L2}w_{L4}) \qquad (4.11)\end{aligned}$$

Equation 4.11 shows that it is possible to have a transformation of the original graph such that edge weight regularization in the original graph is equivalent to node label regularization in the transformation. The proof is by construction.



(a)



(b)

Figure 4.1: The graph at the top is the original graph while the transformed graph is on the bottom. P1, P2 and P3 are objects and K1, K2 and K3 are features. P12, P23, P13 correspond to edges e1, e2 and e3 respectively. K12, K13 and K23 correspond to edges e4, e5 and e6. L1, L2, L3 and L4 are inter-layer edges corresponding to e7, e8, e9 and e10.

The transformed graph, $G'' = (V'', E'')$ is a bipartite graph and is constructed as follows. A node is constructed for every edge in each layer of graph. Let us denote the node corresponding to an edge between u_i and u_j as u_{ij}. There exists an edge between two nodes u_{ij} and v_{kl} if and only if there exist inter-layer edges (u_i, v_k) and (u_j, v_l) or (u_j, v_k) and (u_i, v_l). Note that there exists no edges between nodes of the same layer in the transformed graph and hence, the transformed graph is bipartite. The node labels are initialized to the similarity value of the corresponding edges. The edge between u_{ij} and v_{kl} is weighted as $Z_{u_i,v_k} * Z_{u_j,v_l}$

$$w_{u_{ij},v_{kl}} = \begin{cases} Z_{u_i,v_k} * Z_{u_j,v_l}, & \text{if } \exists \text{ edges } (u_i, v_k) \text{ and } (u_j, v_l) \\ 0 & \text{else} \end{cases} \tag{4.12}$$

- Case 2: Assume the inter-layer edge weights or feature weights are also part of the optimization as in equation 4.1. In this case, consider the dual graph, G'. The node labels corresponding to the inter-layer edges, L_i are also varying. Hence it is not possible to collapse the corresponding nodes as in the previous case. However, we can create two different bipartite graphs, G'' and G''_L. G'' is constructed as before while $G''_L = (V''_L, E''_L)$ is constructed as follows. V''_L is the set of nodes corresponding to inter-layer edges, $L_i \in G'$. There exists an edge between two nodes L_i and L_j if and only if there exists edges

$(L_i, u_{ab}), (u_{ab}, L_j), (v_{cd}, L_i), (L_j, v_{cd}) \in E'$. The weights are assigned as follows,

$$w_{L_i,L_j} = \begin{cases} w_{u_a,u_b} * w_{v_c,v_d}, & \text{if } \exists \text{ edges } (u_a, u_b) \text{ and } (v_c, v_d) \\ 0 & \text{else} \end{cases} \tag{4.13}$$

Thus, the optimization is expressed as a node regularization problem over two decomposed graphs, G'' and G''_L. However, note that after each iteration of node label regularization over the two graphs, the edge weights in both graphs need to be updated using the newly updated node labels.

4.5.3 Computational Complexity

For each iteration, the computation consists of a random walk of length 3 for each node in the graph. Let us assume that the average number of similar neighbors for every node is K and the total number of nodes in all the layers of graphs is N. Therefore the complexity of the algorithm is $O(K^3 * N * t)$, where t is the number of iterations. Note that, the average number of neighbors can be set to a constant if we prune the graph after every iteration to make it a K-nearest neighbor graph. In this case, each iteration runs in linear time and hence, the overall time complexity is $O(N * t)$.

4.5.4 MapReduce Implementation

SMUG can be easily implemented under the MapReduce framework. Each iteration of the algorithm is realized in three MapReduce operations. In the first operation, a record consists of a key node and its neighbors in another layer of graph each attached with the inter-layer edge weight. The mapper outputs a pair of the original key node and the neighbor with the value being the inter-layer edge weight. The second MapReduce operation performs a join operation in the mapper phase with one record being the output of the previous MapReduce operation and the other record being the neighbors of the key node in the same layer. This mapper too outputs a pair of the original key and the neighbor of neighbor as the key with the value being the product of the similarity and the value of the inter-layer edge weight. The third MapReduce performs the reverse of the first MapReduce operation. All the different MapReduce operations use an identity reducer.

4.6 Experiments

It is very hard to evaluate similarity measures in isolation. Thus, most of the algorithms to compute similarity scores are evaluated extrinsically, i.e, the similarity scores are used for an external task like clustering or classification and the performance in the external task is used as the performance measure for the similarity scores. This also helps demonstrate the different applications of the computed similarity measure. Thus, we perform a variety of experiments on standard data sets to illustrate the improved performance of SMUG. There are three natural variants of the algorithm,

- *SMUG*: We compare against SMUG, the edge-weight regularization algorithm described in Chapter III.
- *SMUG-binary*: In this variant, we initialize the importance scores to 1, i.e,

$Z_{ij} = 1$ whenever document i contains the keyword j and zero otherwise.

- *SMUG-TFIDF*: We initialize the importance scores to the TFIDF scores, Z_{ij} is set to the TFIDF score of keyword j for document i.

4.6.1 Experiment I

We compare our similarity measure against other similarity measures in the context of classification. We also compare against a state of the art classification algorithm which uses different similarity measures due to different feature types without integrating them into one single similarity measure. Specifically, we compare SMUG against three other similarity baselines in the context of classification which are listed below.

- *Content Similarity*: Similarity is computed using just the feature vector representation using just the text. We use cosine similarity after preprocessing each document into a $tf \cdot idf$ vector for the AAN data set. For all other data sets, we use the cosine similarity on the binary feature vector representation that is available.

- *Link Similarity*: Similarity is computed using only the links (citations, in the case of publications). To compute link similarity, we use the node similarity algorithm described in (Harel and Koren, 2001) using a random walk of length 3 on the link graph.

- *Linear combination*: The content similarity (CS) and link similarity (LS) between documents x and y are combined in a linear fashion as $\alpha CS(x, y) + (1 -$

$\alpha)LS(x,y)$. We tried different values of α and report only the best accuracy that can be achieved using linear combination of similarity measures. Note that this is a special case of Multiple Kernel Learning (MKL) (Bach et al., 2004). In MKL, the α value is learnt using training data. Note that the number of parameters is linear in the number of different layers.

We also compare SMUG against the following algorithms.

- *SC-MV*: We compare all variants of SMUG against the spectral clustering algorithm for data with multiple views (Zhou and Burges, 2007). The algorithm tries to classify data when multiple views of the data are available. The multiple views are represented using multiple homogeneous graphs with a common vertex set.

  In each graph, the edge weights represent similarity between the nodes computed using a single feature type. For our experiments, we used the link similarity graph and the content similarity graph as described above as the two views of the same data

- *LSA*: Latent semantic analysis (LSA), also known as latent semantic indexing (LSI), can also deal with synonymy and the use of related words. LSA can be used to cluster documents even if they do not use the exact same vocabulary. It makes use of a term-document matrix, X, which describes the occurrences of terms in documents. The $(i,j)^{th}$ entry corresponds to the TF-IDF weight of term i in document j. Clearly, in this matrix, related terms are treated as two different terms. In order to overcome this, LSA tries to find a low-dimensional embedding of the documents so that two terms or documents which are close in the resulting embedding be semantically similar. First, we perform singular value decomposition (SVD) as follows

$$X = U \Sigma V^T \tag{4.14}$$

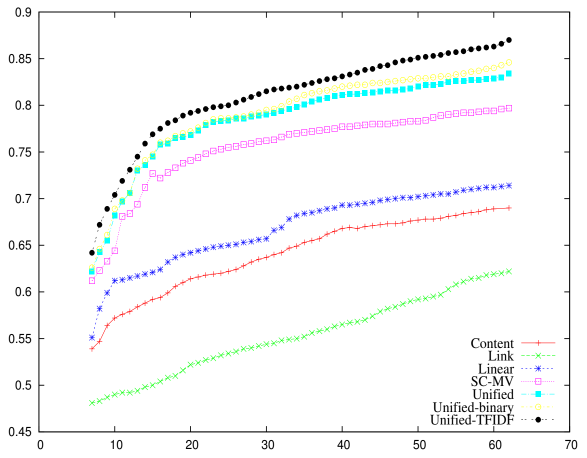
We compute a K-rank approximation to X using the K largest singular values of Σ, using a truncated SVD (Golub and Loan, 1996). Let k be an integer and $k << min(m, n)$.We define U_k to be the first k columns of U, and V_k^T be the first K rows of V^T. We let $\Sigma_k = \mathrm{diag}[\sigma_1, \ldots, \sigma_k]$ contain the first k largest singular values, and define

$$X_k = U_k \Sigma_k V_k^T \tag{4.15}$$

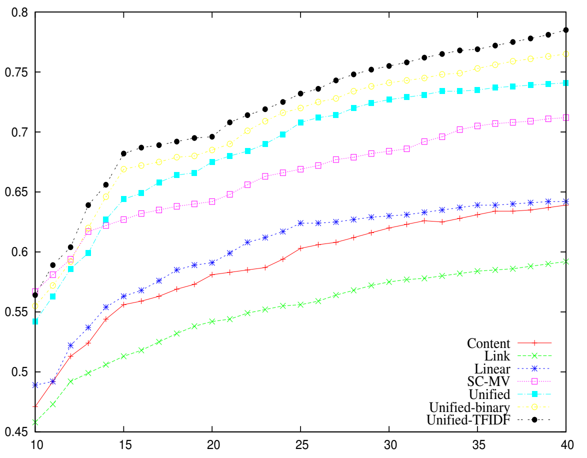
The i^{th} column, d_i in V_k^t corresponds to the representation of the i^{th} document in the latent (concept) space spanned by the k dimensions. Let $\hat{d_i} = \Sigma_k d_i$. Now, the semantic similarity between the i^{th} and the j^{th} document can be computed as follows

$$sim(D_i, D_j) = \sum_{c=1}^{k} \hat{d}_{ic} \hat{d}_{jc} \tag{4.16}$$

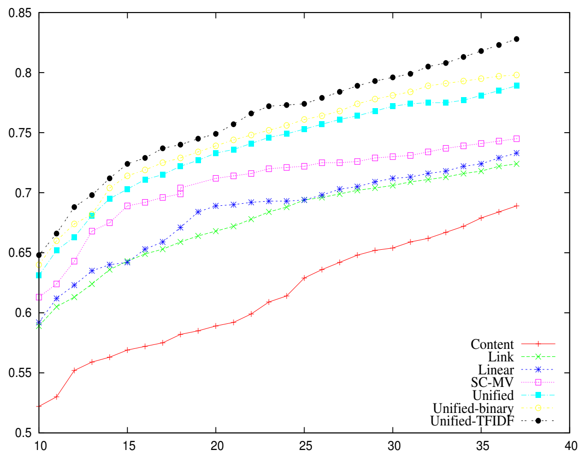
We use the semi-supervised graph classification algorithm (Zhu et al., 2003b) to perform the classification. Refer to Chapter III for a description of the algorithm.



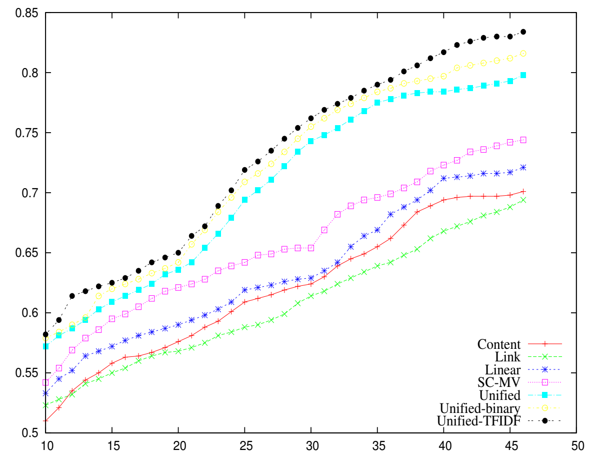
(a) AAN



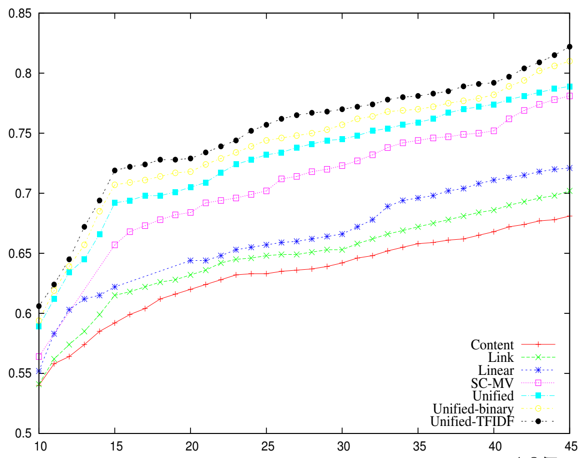
(b) Cornell



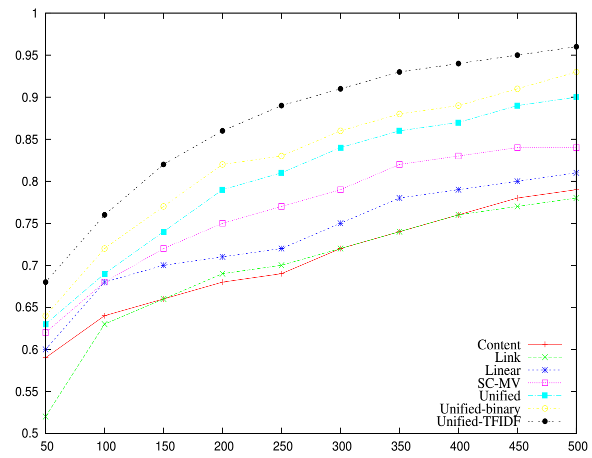
(c) Texas



(d) Washington



(e) Wisconsin



(f) Cora

Figure 4.2: Classification Accuracy on the different data sets. The number of points labeled is plotted along the x-axis and the y-axis shows the classification accuracy on the unlabeled data.

Similarity Measure	AAN	Texas	Wisconsin	Washington	Cornell	Cora
Content Similarity (Cosine)	0.66	0.34	0.42	0.59	0.63	0.48
Link Similarity	0.45	0.49	0.39	0.52	0.56	0.52
Linear Combination	0.69	0.54	0.46	0.54	0.68	0.54
SMUG	0.78	0.69	0.54	0.66	0.72	0.64
SMUG-Binary	0.80	0.68	0.56	0.69	0.74	0.66
SMUG-TFIDF	**0.84**	**0.70**	**0.60**	**0.72**	**0.78**	**0.70**

Table 4.1: Normalized Mutual Information scores of the different similarity measures on the different data sets

Item	Size
Users	1,000,990
Tracks	507,172
Albums	88,909
Genres	992
Artists	27,888
Training Ratings	252,800,275
Validation Ratings	4,003,960
Test Ratings	6,005,940

Table 4.2: Sample Statistics of the Yahoo! KDD Cup 2011 Data Set

4.6.2 Experiment II

We evaluate the performance of SMUG and its variants in the context of clustering. We compare against the three similarity baselines mentioned in section 4.6.1. We use a spectral graph clustering algorithm proposed in (Dhillon et al., 2007) to perform the clustering.

We performed the classification and clustering experiments on the three data sets used in Chapter III.

4.6.3 Experiment III

We illustrate an application of similarity measures using heterogeneous features in the context of recommendation systems. For this purpose, we report results on the recently concluded Yahoo! KDD cup [1]. The task is learning to predict user ratings of musical items. Items can be tracks, albums, artists, or genres. The key difference from other recommendation data sets like Netflix Recommendation data set or Movielens data set, lies in the availability of metadata for computing similarity. We leverage upon the multiple views of the data to compute an unified similarity measure between items.

We reserve special indexing letters for distinguishing users from items: for users u, v, and for items i, j. A rating r_{ui} indicates the preference by user u of item i, where high values mean stronger preference. In this data set, the values are integers ranging from 0 indicating no interest to 100 indicating a strong interest. We distinguish predicted ratings from known ones, by using the notation $\hat{r}_{ui}$ for the predicted value of r_{ui}. The (u, i) pairs for which r_{ui} is known are stored in the set $\kappa = (u, i) | r_{ui}$ is known. A large majority (99.99%) of the possible ratings are unknown in the KDD cup dataset. Table 4.2 shows some statistics of the KDD cup dataset.

The most common approach to collaborative filtering is neighborhood models (Koren, 2008). We use an item-based neighborhood model instead of an user-based model to predict the ratings because of ease of computation. The basic hypothesis in item-oriented neighborhood models is that similar items receive similar ratings.

We create the following sets of graphs:

[1] `http://kddcup.yahoo.com`

1. Rating Similarity Graphs (RSG): Similarity due to ratings is computed based on the Pearson correlation coefficient ρ_{ij}, which measures the tendency of users to rate items i and j similarly. The Pearson correlation coefficient between two items is computed as follows,

$$\rho_{ij} = \frac{\sum_{u=1}^{\eta ij}(r_{ui} - \bar{r_i})(r_{uj} - \bar{r_j})}{\sqrt{\sum_{u=1}^{\eta_i}(r_{ui} - \bar{r_i})^2}\sqrt{\sum_{u=1}^{\eta_j}(r_{uj} - \bar{r_j})^2}} \tag{4.17}$$

where η_{ij} denotes the number of users who rated both i and j and η_i is the number of users who have rated item i. Also $\bar{r_i}$ is the mean rating of item i. The above correlation coefficient gives a value between -1 and 1. Since many ratings are unknown, it is expected that some items share only a handful of common raters. Computation of the correlation coefficient is based on the common user support. Accordingly, similarities based on a greater user support are more reliable. Therefore, We use the similarity measure as defined in (Koren, 2008) as follows,

$$S_{ij} = \frac{\eta_{ij}}{\eta_{ij} + \lambda_1}\rho_{ij} \tag{4.18}$$

The value of λ_1 is found by cross-validation and is set to 150 in our experiments. Note that if there are no common raters, the similarity is equal to 0.

Now, we can create the following graphs,

- Album Similarity Graph: $G_b = (V_b, E_b)$, where $V_b = \{b_1, b_2, \ldots, b_{n_b}\}$ is the set of albums and n_b is the number of albums.

- Artist Similarity Graph: $G_a = (V_a, E_a)$, where $V_a = \{a_1, a_2, \ldots, a_{n_a}\}$ is the set of artists and n_a is the number of artists
- Genre Similarity Graph: $G_g = (V_g, E_g)$, where $V_g = \{g_1, g_2, \ldots, g_{n_g}\}$ is the set of artists and n_g is the number of genres.
- Track Similarity Graph:$G_t = (V_t, E_t)$, where $V_t = \{t_1, t_2, \ldots, t_{n_t}\}$ is the set of artists and n_t is the number of tracks.

In all of the above graphs, the edge weights are initialized with the similarity function described in Equation 4.18 and there exists an edge between two items in any graph if and only if the two items are among the K most similar items of each other.

2. Genre Similarity Graphs (GSG): We compute similarity between items in an intrinsic fashion using the available metadata, specifically, the genres associated with different items. For every track and album, the metadata contains an ordered list of genres. The weight of genre for a track or album is computed as follows,

$$w_{g_i,item} = \frac{1}{\exp^{-k}} \tag{4.19}$$

where g_i is the k^{th} genre in the ordered list of genres for *item*.

Though the metadata does not contain genre information for artists, we create a genre vector for each artist by aggregating the list of genres for the artist's albums and tracks. The weight of a particular genre is equal to the frequency count in the aggregated list of genres.

We use the following graphs to compute similarity between items, $G_{bg} = (V_b, E_{bg})$, $G_{ag} = (V_b, E_{ag})$ and $G_{tg} = (V_t, E_{tg})$. The edge weights in all these graphs are initialized to the cosine similarity between the genre vectors of the corresponding items. We use G_g for genre similarity.

The edges across the different graphs are initialized using the metadata. There exists an edge between two different items if they are related due to the metadata. For example, there exists an edge between a track and a genre, if the track belongs to the particular genre according to the metadata. All the inter-layer edges are unweighted.

3. Combined Similarity Graphs (CSG): We combine the above two sets of graphs by simply setting the similarity between the items to the mean of rating similarity and genre similarity.

We ran SMUG and SMUG-binary on all the above sets of graphs to unify the different graphs.

The goal is to predict r_{ui} − the unobserved rating by user u for item i. Using the similarity measure, we identify the K items rated by u, which are most similar to i. This set of k neighbors is denoted by $S_k(i;u)$. The predicted value of r_{ui} is taken as a weighted average of the ratings of neighboring items, while adjusting for user and item effects through the baseline estimates as follows,

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in S_k(i;u)} Sim_{ij}(r_{uj} - b_{uj})}{\sum_{j \in S_k(i;u)} Sim_{ij}} \tag{4.20}$$

where b_{ui} is a baseline rating estimate for item i by user u. We refer to the recommendation system described by equation 4.20 as *Indentity.* Let μ be the overall average rating. The baseline rating is computed as

$$b_{ui} = \mu + b_i + b_u \tag{4.21}$$

The parameters b_u and b_i indicate the observed deviations of user u and item i, respectively, from the average. For example, suppose that we want a baseline estimate for the rating of the song “Hotel California” by user X. Now, say that the average score over all tracks, μ, is 58. Furthermore, “Hotel California” is better than an average track, so it tends to be rated 15 stars above the average. On the other hand, X is a critical user, who tends to rate 10 points below the average. Thus, the baseline estimate for “Hotel California”’s rating by $X = 58 + 15 - 10 = 63$.

We compute b_u and b_i by solving the following least squares problem

$$\sum_{u,i \in \kappa} (r_{ui} - \mu - b_u - b_i)^2 + \lambda_2 (\sum_u b_u^2 + \sum_i b_i^2) \tag{4.22}$$

Let the test set be denoted by TS. We use Root Mean Squared Error (RMSE) to measure the quality of predicted ratings. RMSE is computed as follows

$$RMSE = \sqrt{\sum_{u,i \in \mathrm{TS}} (r_{ui} - \hat{r}_{ui}{}^2)/|\mathrm{TS}|} \tag{4.23}$$

Algorithm	Similarity Source		
	RSG	**GSG**	**CSG**
Identity	26.78	27.92	26.50
SMUG	25.54	26.50	25.44
SMUG-Binary	25.12	26.14	24.81

Table 4.3: RMSE scores obtained by the different similarity measures on the 2011 KDD cup test set

4.7 Results and Discussion

Figure 4.2 shows the accuracy of the classification obtained using different similarity measures. It can be seen that SMUG (both the variants) performs much better than other similarity measures by a large margin. SMUG-TFIDF performs better than all other variants. For every different size of the training set, we chose the training set randomly 5 times. For example, for a training set of size 10, we chose 10 points randomly 5 times and averaged the five accuracy values to obtain a single accuracy value corresponding to a training set size 10. We also conducted $1-$tailed t tests with the following null hypotheses: The baselines/other algorithms' classification accuracy is better than our algorithm's classification accuracy. All the hypotheses are rejected with a p-value of 0.01 which means our algorithm's classification accuracy is higher than all other algorithms' classification accuracy and the result is statistically significant at the 99% confidence level. The algorithm performs much better when more information is provided in the form of TF-IDF scores. We attribute this to the rich representation of the data. In all variants of SMUG, the data is represented as a set of heterogeneous graphs (layers) which are connected together instead of the feature vector representation. Thus, we can leverage using the similarity between the features (keywords) and the objects (documents) to iteratively improve similarity in both layers. Whereas, in the case of the algorithm in (Zhou and Burges, 2007) all the graphs are isolated homogeneous graphs. In other words, there are two graphs created with the same set of vertices. Each graph is created by a different similarity measure. The spectral clustering algorithm tries to find a clustering which is close to optimal for both the graphs. However, each feature space similarity could result in many different graphs. For example, consider the following scenario. There are four publications: p_1, p_2, p_3, p_4. Let publications p_1 and p_2 be similar according to content similarity while p_1 and p_3 are similar according to link similarity. In this case, the spectral clustering algorithm cannot cluster the two graphs because it does not merge the information from the two graphs before attempting to cluster the two graphs. Hence there is no information transfer across the different graphs.

In an attempt to understand how the topology of the graphs relates to the classification accuracy, we computed the clustering coefficient of the different graphs. We computed the Newman Clustering Coefficient (NCC) (Newman, 2003c) as follows,

$$NCC = \frac{3 * \text{number of triangles in the network}}{\text{number of connected triples in the network}} \tag{4.24}$$

Table 4.4 shows the clustering coefficient scores of the different graphs. It can be seen that the classification accuracy obtained by the different similarity measures is proportional to the clustering coefficient. We also computed the Pearson correlation coefficient between the clustering coefficient of a similarity graph and the normalized mutual information obtained using the particular graph. The correlation coefficient lies between 0.77 and 0.79 for the content similarity graph and the link similarity

Similarity Measure	**AAN**	**Texas**	**Wisconsin**	**Washington**	**Cornell**	**Cora**
Content Similarity (Cosine)	0.54	0.29	0.38	0.54	0.56	0.4
Link Similarity	0.41	0.36	0.36	0.48	0.50	0.45
Linear Combination	0.63	0.52	0.45	0.52	0.63	0.5
SMUG	0.69	0.63	0.50	0.57	0.69	0.61
SMUG-Binary	0.75	0.65	0.54	0.63	0.68	0.63
SMUG-TFIDF	**0.79**	**0.69**	**0.60**	**0.69**	**0.72**	**0.68**

Table 4.4: Clustering Coefficient of the different similarity graphs on the different data sets

graph across all datasets. However, the clustering coefficient of the SMUG-induced graphs is more correlated with the NMI values with a correlation coefficient of 0.90. This shows that SMUG does improve the clustering coefficient of the similarity graphs by using the relation (edges) between the graphs.

Table 4.1 shows the NMI scores obtained by the different similarity measures on the different data sets. Table 4.3 shows the RMSE scores obtained by the different similarity measures on the KDD cup test set.

We use two different feature types to compute the similarity between items in the recommendation task: user ratings and genres. Every item is associated with a set of genres and user ratings. The edge weights in the RSG graphs represent similarity computed using user ratings while the edge weights in GSG graphs represent similarity computed using genre information. It can be seen that all the algorithms perform better when the similarity between items is computed using user ratings instead of genres. This is because there is a large number of available ratings to compute similarity between items. Hence, it is possible to obtain a reliable similarity estimate between a large number of items. However, in the case of computing similarity using genres, there is a large number of items which use a few specific genres. Hence, the similarity computed between the items is artificially high.

Nevertheless, the combined similarity graphs perform the best irrespective of the similarity learning algorithm. This shows that there is some amount of information contained in the genres that cannot be extracted from the user ratings. Note that, both variants of SMUG outperform the baseline system irrespective of the set of graphs being used and SMUG-binary performs the best.

CHAPTER V

Topic Detection in Blogs

5.1 Introduction

In chapters 3 and 4, we show how the proposed framework can be used to integrate multiple similarity measures by learning from one another. In this chapter, we present another example application of the proposed framework in chapter 3, topic detection. Topic detection is the problem of nding the set of most prominent topics in a collection of documents. We define a topic as a cohesive collection of documents discussing the same topic. We use keyphrases to represent topics. For example, consider the "Virginia Tech Shootings" event [1], the keyphrase "Gun Ownership" represents a topic since the documents containing the keyphrase are a cohesive set of documents. Given a set of documents about a particular event, the goal is to find a set of representative and discriminative (non-redundant) topics in the given set of documents. We create two connected layers of graphs. The first graph is a keyphrase graph, where the nodes

[1]The Virginia Tech massacre was a school shooting that took place on April 16, 2007, on the campus of Virginia Polytechnic Institute and State University in Blacksburg, Virginia, United States.

correspond to keyphrases and the edge weight represents the similarity between the keyphrases. The second graph is a document graph where each node represents a document and the edges correspond to document similarity. There is an edge between a document and a keyphrase if the document contains the keyphrase. We formulate the task of selecting representative keyphrases as a weighted set cover problem defined over the two layers of graphs.

Finding a list of topics that a collection of documents covers is an important problem in information retrieval. These topics can be used to describe or summarize the collection, or they can be used to cluster it. Topics also provide a short and informative description of the documents that can be used for quickly browsing and finding related documents. In essence, a collection of documents has an underlying set of topics and we want to identify the topics by clustering related topics and documents. Individual documents can contain multiple topics.

Traditionally, information retrieval systems return a ranked list of documents based on the IR similarity to the user's query. Unfortunately, the results returned are often redundant. Users may need to reformulate their search to find the specific topic they are interested in. This active searching process leads to inefficiencies, especially in cases where queries or information needs are ambiguous. For example, a user wants to get an overview of the virginia tech shootings, then the first query he/she might try is "virginia tech shooting". Most of the results returned would be posts just mentioning the shootings and the death toll. But the user might want a more detailed overview of the shootings. Thus this leads to continuously reformulating the search query to discover all the topics in the event.

5.2 Related Work

Topic detection and tracking was studied extensively on newswire and broadcast collections by the NIST TDT research program (Allan et al., 2003). The large number of people blogging on the web provides a new source of information for topic detection and tracking.

The TDT task defines topics as "an event or activity, along with all directly related events and activities." In this work we will stay with this definition of topic. (Zhai et al., 2003) proposed several methods for dealing with a related task, which they called *topic retrieval* This is an information retrieval task where the goal is to retrieve and return documents that cover the different topics of a given query. As they point out, the utility of each document is dependent on the other documents in the ranking, which violates the independent relevance assumption traditionally used in IR.

To reduce the complexity of this task, a candidate set of topics needs to be generated that cover the document collection. We choose to use a keyphrase detection algorithm to generate topic labels. Several keyphrase extraction algorithms have been discussed in the literature, including ones based on machine learning methods (Turney, 2000), (Hulth, 2003) and tf-idf (Frank et al., 1999). Our method uses language models and pointwise mutual information expressed as the Kullback-Leibler divergence.

Kullback-Leibler divergence has been found to be an effective method of finding keyphrases in text collections (Tomokiyo and Hurst, 2003). But identification of keyphrases is not enough to find topics in document. The keyphrases identified may describe the entire collection, or aspects of the collection.

The problem of topic detection is also related to novelty detection in (Allan et al., 2003). In this problem, given a set of previously seen documents, the task is to determine whether a new document contains new or novel content. A document is considered not novel or "redundant" if all of the relevant information in the document is covered by relevant documents delivered previously. Note that this definition includes "duplicate" and "near duplicate" documents as well as documents that are redundant in content but very different in presentation. In the TREC 2002 novelty track, the task was to discard sentences that did not contain new material. This is similar to our goal of reducing redundancy in the list of returned topics.

In most cases, novelty detection is implemented as an online algorithm. The system has a set of existing documents they have seen up until a certain point. The task is to determine whether a new document is novel based on the previous documents. Once a decision has been made, the label of that document is fixed. Mathematically, the approach can be formulated as follows,

- d_t: a document that arrives at time t and that is being evaluated for redundancy.
- $D(t)$: the set of all documents delivered to the user by the time d_t arrives, not including d_t.
- $DR(t)$: the set of all relevant documents delivered till time t. $DR(t) \subset D(t)$.
- $R(d_t)$: the measure of redundancy for document d_t.
- d_i: usually refers to a relevant document that was delivered before d_t arrived.

The redundancy of document d_t depends on the set of previously seen documents, $D(t)$. Specifically, it depends on the set of previously seen relevant documents, $DR(t)$. Therefore, $R(d_t) = R(d_t|DR(t))$

There is a subtle difference between novelty detection and topic detection in that topic detection is an offline task. Therefore, the topic detection algorithm typically has access to the entire document set.

5.3 Existing redundancy measures

(Zhang et al., 2002) examine five different redundancy measures for information filtering. Given a document stream, the task of information filtering is to return the documents relevant to a user. Examples of information filtering systems include traditional information retrieval systems that return relevant documents depending on the user's query. Adaptive information filters change their behavior based on changes in the document stream or feedback from the user.

The redundancy measures examine are based on online analysis of documents. They identify two different methods of measuring redundancy:

- Let i documents in the stream have been processed and clustered into k clusters. In order to process the $i+1^{st}$ document, we compute the similarity of the $i+1^{st}$ document to each of the k clusters and add the document to the closest cluster if the similarity is above a threshold, else we create a new cluster with only the $i+1^{st}$ document.

- Measure similarity between the new document and each previously seen document and label the document as redundant if the maximum similarity to any previously seen document is above a threshold, i.e, $R(d_t|DR(t)) = argmax_{d_i \in DR(t)} R(d_t|d_i)$

(Zhang et al., 2002) compare the following measures of redundancy.

- Set difference: Set difference measures the novelty of a document using the number of new words that occur in the new document compared to the previous documents. If a word w_i occurred frequently in document d_t but less frequently in an old document d_i, it is likely that new information not covered by d_i is covered by d_t They smooth word counts using word counts from all previously seen documents and previously seen relevant documents to account for stop words and topic-specific stop words respectively.

- Geometric distance: Geometric distance refers to distance metrics between documents represented as vectors, such as Euclidean distance and cosine similarity.

- Distributional similarity: Distributional similarity refers to computing similarity between language models using probabilistic distribution similarity measures such as KL divergence. They use two different smoothing methods in order to account for rare, unseen words.

  - Bayesian smoothing with Dirichlet priors: Language models are multinomial distributions. The conjugate prior for a multinomial distribution is the Dirichlet distribution, which can be used to smooth the language models (MacKay and Peto, 1994).

  - This approach smooths by shrinking parameter estimates in sparse data towards the estimates in rich data. For estimating the language model of document d, we can shrink its MLE estimator θ_{d_MLE} with the MLE estimator of a language model for general English θ_{E_MLE} and the MLE estimator of a language model for the topic θ_{T_MLE}:

$$\theta_d = \lambda_d \theta_{d\_MLE} + \lambda_T \theta_{T\_TMLE} + \lambda_E \theta_{E\_MLE} \tag{5.1}$$

where $\lambda_d + \lambda_T + \lambda_E = 1$

- Mixture model: They also propose a new algorithm based on generative models of document creation. Their algorithm combines language models with KL divergence. Their model assumes documents are generated by three language models: a general English language model, a topic model, and an individual document-specific information model. They find coefficients for each term by using the EM algorithm.

They evaluated the systems on the dataset obtained by combining AP News and Wall Street Journal data from TREC CDs 1, 2, and 3. The relevance judgments for the dataset are available from NIST. They found that cosine similarity was the most effective measure with an F1-score of 0.625, followed by the new mixture model measure (F1-score: 0.61).

5.4 Data

The data was collected from the Blogocenter bloglines database The Blogocenter group at UCLA has been retrieving RSS feeds from the Bloglines, Blogspot, Microsoft Live Spaces, and syndic8 aggregators for the past several years.

Event	**Description**	**Posts**	**Dates**
iphone	iPhone release hype	48810	June 20, 2007 - July 7, 2007
petfoodrecall	Recall of Melamine tainted petfood	4285	March 10, 2007 - May 10, 2007
spitzer	Eliot Spitzer prostitution scandal	10379	March 6, 2008 - March 23, 2008
vtech	Virginia Tech shooting	12256	April 16, 2007 - April 30, 2007

Table 5.1: Major events summarized

> After the Virginia Tech murders, there's the usual outcry for something to be done, and in particular, for more gun control. As usual, I am not persuaded. The Virginia Tech campus had gun control, which meant that Cho Seung-Hui was in violation of the law even before he started shooting, and also that no law-abiding citizens were able to draw ...

Figure 5.1: Example blog post discussing video games (Hoopdog, 2007)

We choose four news events that occurred in 2007 and 2008 based on our expectation of discussion level in the blogosphere. Table 5.1 shows the number of posts collected for the chosen events.

We will use the Virginia Tech shooting as a running example. People throughout the blogosphere posted responses expressing support and condolences for the people involved, along with their own opinions on what caused the rampage.

Figures 5.1 and 5.2 show two different responses to the event. The quote in figure 5.1 shows an example post from LiveJournal, a popular blogging community. In this post, the user is discussing his view on gun control, a hotly debated topic in the aftermath of the shooting. Figure 5.2 expresses another person's emotional response to this event. Both posts show different aspects of the same story. Our topic detection system seeks to automatically identify these and other distinct discussions that occur around an event.

Figure 5.3 shows a generalized Venn diagram (Kestler et al., 2005) of the cluster

... Predictably, there have been rumblings in the media that video games contributed to Cho Seung-Hui's massacre at Virginia Tech. Jack Thompson has come out screaming, referring to gamers as “knuckleheads” and calling video games “mental masturbation” all the while referring to himself as an “educator” and “pioneer” out to “right” society. ...

Figure 5.2: Example blog post discussing video games (Hoopdog, 2007)

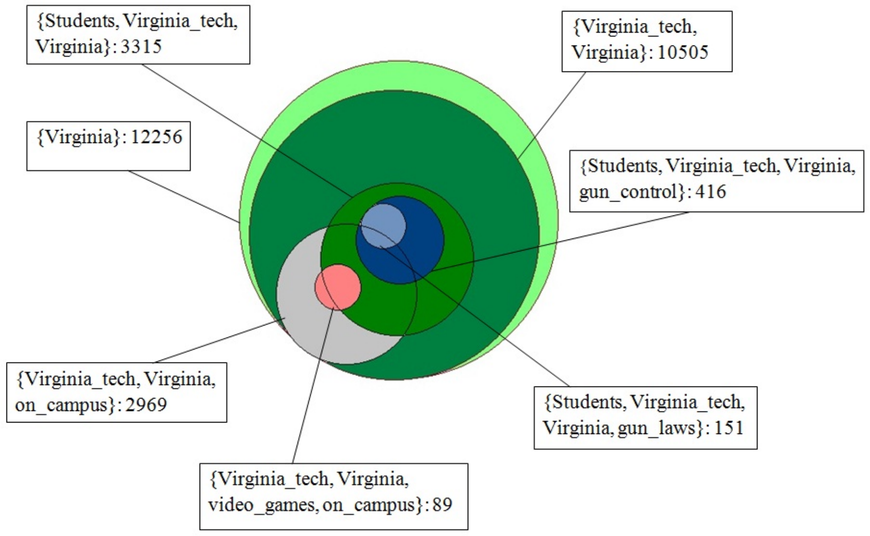

{Students, Virginia_tech, Virginia}: 3315
{Virginia_tech, Virginia}: 10505
{Virginia}: 12256
{Students, Virginia_tech, Virginia, gun_control}: 416
{Virginia_tech, Virginia, on_campus}: 2969
{Students, Virginia_tech, Virginia, gun_laws}: 151
{Virginia_tech, Virginia, video_games, on_campus}: 89

Figure 5.3: Generalized Venn diagram of topic overlap in the Virginia Tech collection

overlap between different keyphrases from the Virginia Tech event.

Preprocessing: For each news item, relevant posts were retrieved, based on keyword searching and date of blog post. Posts from the date of occurrence of the event to the day when less than 50 relevant blog posts were posted were gathered. We removed all posts which had less than 5 sentences of text.

5.5 Method

The task is to find discriminative labels for the different topics that exist in a collection of documents. Taken together, these labels should satisfy the following conditions:

- Describe a large portion of the collection
- The overlap between the topics should be minimal

This problem is similar to Minimum Set Cover, which is NP-complete (Garey and Johnson, 1990). Therefore, trying to find the optimal solution by enumerating all possible phrases in the corpus would be impossible, instead we propose a two-step method for topic detection.

The first step is to generate a list of candidate phrases. These phrases should be informative and representative of all of the different topics. The second step should select from these phrases consistent with the two conditions stated above.

5.5.1 Generating Candidate Phrases

We want to generate a list of phrases that have a high probability of covering the document space. There are many methods that could be used to find informative keyphrases. One such method is using the standard information retrieval TF-IDF model (Jones, 1972). Another method is using Kullback-Leibler divergence.

(Tomokiyo and Hurst, 2003) developed a method of extracting keyphrases using statistical language models. They considered keyphrases as consisting of two features, *phraseness* and *informativeness*. Phraseness is described by them as the "degree to which a given word sequence is considered to be a phrase." For example, collocations have a high phraseness. Informativeness is the extent to which a phrase captures the key idea or main topic in a set of documents.

To find keyphrases, they compared two language models, the target document set and a background corpus. Pointwise KL divergence was chosen as the method of finding the difference between two language models.

The KL divergence $D(p||q)$ between two probability mass functions $p(x)$ and $q(x)$ with alphabet χ is given in equation 5.2.

$$D(p||q) = \sum_{x \in \chi} p(x) log \frac{p(x)}{q(x)} \tag{5.2}$$

KL divergence is an asymmetric function. $D(p||q) \neq D(q||p)$.

Pointwise KL divergence is the individual contribution of x to the loss of the entire distribution. The pointwise KL divergence of a single phrase w is $\delta_w(p||q)$:

$$\delta_w(p||q) = p(w) log \frac{p(w)}{q(w)} \tag{5.3}$$

The phraseness of a phrase can be found by comparing the foreground n-gram language model against the background unigram model. For example, if we were judging the phraseness of "gun control", we would find the pointwise KL divergence of "gun control" between the foreground bigram language model and the foreground unigram language model.

$$\varphi_p \delta_w(LM_f g^N || LM_f g^1) \tag{5.4}$$

The informativeness of a phrase can be found by finding the pointwise KL divergence of the foreground model against the background model.

$$\varphi_i = \delta_w(LM_f g^N || LM_b g^N) \tag{5.5}$$

A unified score can be formed by adding the phraseness and informative score:

$$\varphi = \varphi_p + \varphi_i \tag{5.6}$$

Once keyphrases have been extracted from the document set, they are sorted based on their combined score. We select the top n-ranked keyphrases as candidate phrases.

Based on our chosen task conditions regarding coverage of the documents and minimized overlap between topics, we need an undirected mapping between phrases and documents. A natural representation for this is a bipartite graph where the two sets of nodes are phrases and documents. Let the graph be: $G = (W, D, E)$ where W is the set of candidate phrases generated by the first step and D is the entire set of documents. E is the set of edges between W and D where there is an edge between a phrase and a document if the document contains the phrase.

We formulate the task as a variation of Weighted Set Cover problem in theoretical computer science. In normal Set Cover we are given a collection of sets S over a universe U, and the goal is to select a minimal subset of S such that the whole universe, U is covered. Unfortunately this problem is NP-complete (Garey and Johnson, 1990), so we must settle for an approximate solution. But fortunately there exist very good

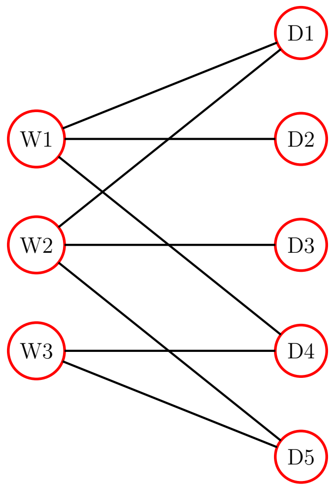


Figure 5.4: Bipartite graph representation of topic document coverage

α-approximation algorithms for this problem (Cui, 2007).

The difference between weighted set cover and set cover is that each set has an associated real-valued weight or cost and the goal is to find the minimal cost subset which covers the universe U.

In our problem, each phrase can be thought of as a set of the documents which contain it. The universe is the set of all documents.

5.5.2 Greedy Algorithm

To solve the above problem, we use a greedy algorithm. which computes a cost for each node iteratively and selects the node with the lowest cost at every iteration. The cost of a word should be such that we do not choose a phrase with very high coverage, like "virginia" and at the same time not choose words with very low document frequency since a very small collection of documents can not be judged a topic.

Based on these two conditions we have come up with a linear combination of three cost factors, similar to Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998).

1. Relative Document Size:

   $$f_1(w_i) = \frac{|adj(w_i)|}{N} \tag{5.7}$$

   where $adj(w_i)$ is the set of documents that the keyphrase w_i occurs in. $|adj(w_i)|$ is the document frequency of the word.

   This factor takes into account that we do not want to choose words which cover the whole document collection. For example, phrases such as "virginia" or "virginia tech" are bad topics, because they cover most of the document set.

2. Redundancy Penalty:

   We want to choose elements that do not have a lot of overlap with other elements. One measure of set overlap is the Jaccard similarity coefficient:

   $$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{5.8}$$

   $$f_2(w_i) = 1 - \frac{\sum_{w_j \in W - w_i} J(w_i, w_j)}{|W| - 1} \tag{5.9}$$

   This factor is essentially $1-$ average jaccard similarity.

We calculate the pairwise Jaccard coefficient between the target keyphrase and every other keyphrase. The pairwise coefficient vector provides information on how much overlap there is between a keyphrase and every other keyphrase. Phrases with a high average Jaccard coefficient are general facets that cover the entire collection. Phrases with a low Jaccard coefficient are facets that cover specific topics with little overlap.

3. Subtopic Redundancy Memory Effect

   Once a keyphrase has been chosen we also want to penalize other keyphrases that cover the same content or document. Equation 5.10 represents a redudancy "memory" for each keyphrase or topic. This memory is updated for every step in the greedy algorithm.

$$R(w_i) = R(w_i) + J(w_i, w_j) \tag{5.10}$$

   where w_j is the newly selected phrase.

A general cost function can be formed from a linear combination of the three cost factors. We provide two parameters, α and β to reprsent the tradeoff between coverage, cohesiveness and intersection. For our experiments, we found that an α value of 0.7 and a β value of 0.2 performed well.

$$
\begin{aligned}
cost(w_i) = {} & \alpha \times f_1(w_i) \\
& +\beta \times f_2(w_i) \\
& +(1-(\alpha+\beta)) \times R(w_i)
\end{aligned} \tag{5.11}
$$

The pseudocode for the greedy algorithm is given in Figure *A.1*.

Algorithm *A.1*

(∗ A greedy set-cover algorithm for detecting sub-topics ∗)

Input: Graph $G = (W, D, E)$, N: number of documents to cover

Output: Set of discriminative phrases for the different topics

```
1.  $W = \{w_1, w_2, \ldots, w_n\}$
2.  $W_{chosen} = \emptyset$
3.  $num\_docs\_covered = 0$
4.  while $num\_docs\_covered < N$
5.      do for $w_i \in W$
6.              do $cost(w_i) = \alpha \times f_1(w_i)$
7.                  $+\beta \times f_2(w_i)$
8.                  $+(1-(\alpha+\beta)) \times R(w_i)$
9.          $w_{selected} = \operatorname{argmax}_w cost(w_i)$
10.         for $w_i \in W$
11.             do $R(w_i) = R(w_i) + J(w_{selected}, w_i)$
12.         $num\_docs\_covered = num\_docs\_covered + |adj(w_{selected})|$
13.         $W_{chosen} = W_{chosen} \cup \{w_{selected}\}$
```

```
14.         $W = W - \{w_{selected}\}$
15.         $D = D - adj(selected)$
16. return $W_{chosen}$
```

5.6 Adaptive Algorithm

There are two problems with the above greedy algorithm. First, there is no easy way to tell how many documents should be covered so that the set of topics chosen are distinct and informative. Secondly, even if there is an oracle to tell us the right number of documents to be covered, identifying which documents to be covered is an issue. Not all documents are equally good. For example, in the *Virginia Tech Tragedy* example, there are a lot of documents which merely report the tragedy and the death toll. These documents are good candidates to be covered if the number of topics required by the user is small. Therefore, the utility of a document depends on the number of topics required by the user. If the user requires just two topics, then good representative terms are "Virginia Tech Tragedy" and "mass shooting". However, the same two terms are not good representative topics if the number of topics requested by the user is, say 20. This suggests that we need a scoring function for the documents which depends on the number of topics to be chosen and the scores of the keyphrases it contains. The score for each keyphrase would still be computed using the same function as in the greedy algorithm, but each factor would be a weighted sum with the weights being the scores of the documents. Thus, the algorithm is recursive with the scores of the words being computed using the scores of the documents, which is in turn computed using the scores of the keyphrases.

The scoring function should be such that when the number of topics requested by the user is small, then the scoring function should prefer documents which contain a lot of high-scoring keyphrases and prefer keyphrases which cover a lot of high-scoring documents. When the number of topics requested is low, then the scoring function should prefer documents which are focussed, or in other words, contain few keyphrases and prefer keyphrases, which are focussed topics, or in other words, are contained in a few number of documents. Therefore we iteratively compute the scores of keyphrases and documents until they converge. After each iteration, we normalize the score vectors of keyphrases and documents so that they sum to 1.

Let the score of the keyphrase w_i be W_score_i and the score of document d_j be D_score_j. Initially we set the score of all documents to $1/N$ where N is the total number of documents to be covered. Let the score vector for keyphrases and documents be denoted as W_score and D_score respectively. The weighted version of the formulae for the relative document size factor and the redundancy penalty is shown below.

$$f_1'(w_i) = \sum_{d_k \in adj(w_i)} D\_score_k \tag{5.12}$$

$$f_2'(w_i) = 1 - \frac{\sum_{w_j \in W - w_i} J'(w_i, w_j)}{|W| - 1} \tag{5.13}$$

where $J'(w_i, w_j)$ is the weighted jaccard similarity coefficient between keyphrases w_i and w_j as given below.

$$J'(A,B) = \frac{\sum_{d_k \in adj(w_i) \cap adj(w_j)} D\_score_k}{\sum_{d_k \in adj(w_i) \cup adj(w_j)} D\_score_k} \tag{5.14}$$

The formula for the topic redundancy memory remains the same. Now the scoring function for the keyphrase is

$$\begin{aligned} W\_score_i = {} & \alpha \times e^{|\frac{K}{max\_topics} - f_1'(w_i)|} \\ & + \beta \times f_2'(w_i) \\ & + (1 - (\alpha + \beta)) \times R(w_i) \end{aligned} \tag{5.15}$$

where K is the number of topics remaining to be chosen and max_topics is the maximum number of topics the user might request. In our experiments, we set this value to 30. The scores for the documents is calculated using the following formula,

$$D\_score_i = e^{|\frac{K}{max\_topics} - g(d_i)|} \tag{5.16}$$

where $g(d_i)$ is the sum of the scores of the keyphrases the document contains, as shown below.

$$g(d_i) = \sum_{w_j \in adj(d_i)} W\_score_j \tag{5.17}$$

The adaptive algorithm is given in *A.2*.

Algorithm *A.2*

($*$ A greedy set-cover algorithm for detecting sub-topics $*$)

Input: Graph $G = (W, D, E)$,K: number of topics to choose

Output: Set of K discriminative phrases for the different topics

```
1.  $W = \{w_1, w_2, \ldots, w_n\}$
2.  $W_{chosen} = \emptyset$
3.  $num\_topics\_to\_cover = K$
4.  for $i \leftarrow 1$ to $m$
5.      do $D\_score_i = \frac{1}{m}$
6.  while $num\_topics\_to\_cover > 0$
7.          repeat
8.              $(W'\_score, D'\_score) = Update\_Scores(W\_score, D\_score)$
9.          until $(||W\_score - W'\_score||_2 \leq \epsilon$ and $||D\_score - D'\_score||_2 \leq \epsilon)$
10.         $w_{selected} = \operatorname{argmax}_{w} cost(w_i)$
11.         for $w_i \in W$
12.             do $R(w_i) = R(w_i) + J'(w_{selected}, w_i)$
13.         $num\_topics\_to\_cover = num\_topics\_to\_cover - 1$
14.         $W_{chosen} = W_{chosen} \cup \{w_{selected}\}$
15.         $W = W - \{w_{selected}\}$
16.         $D = D - adj(selected)$
17. return $W_{chosen}$
```

5.7 Experiments

There are two phases in our algorithm, the first phase consists of extracting keyphrases from the corpus of documents and the second phase consists of pruning the set of keyphrases which can be used to represent topics. As a baseline measure, we implemented a very simple algorithm similar to the Maximal Marginal relevance method (Carbonell and Goldstein, 1998). The algorithm is shown in *A.3* As a gold standard, we manually annotated four different collections of blog posts. Each annotator generated a list of topics.

Algorithm *A.3*

(∗ A simple algorithm based on Maximal Marginal Relevance for finding topics ∗)

Input: T Ranked list of keyphrases,k number of topics required

Output: k discriminative keyphrases for the different topics

```
1.   $L = \emptyset$
2.   $U = T$
3.   $ctr = 0$
4.   while $ctr < k$
5.        $t = argmax_x score(x)$
6.        $L = L \bigcup t$
7.        $U = U - t$
8.        for $x \in U$
9.             $score(x) *= (1 - Sim(x, t))$
10.            $ctr++$
11.       return $L$
```

5.8 Evaluation

We compare our weighted set cover approach for extracting topics with the current state-of-the-art topic modeling approach using Latent Dirichlet Allocation (LDA)

(Blei et al., 2003). The topic model based on LDA, like all other topic models, has the same fundamental idea that a document is a mixture of topics. It is a generative model for documents: it specifies a probability distribution over topics and then, for word in a document, a topic is chosen at random according to this distribution and then a word from that topic is chosen. Using approximate inference technics, this process is inverted and topics responsible for generating this set of documents are inferred. Let $P(z)$ represent the probability distribution over topics in a particular document and $P(w|z)$ for the probability distribution of words over the given topic, z. Then let $P(z_i = j)$ represent the probability that the j^{th} topic is chosen for the i^{th} term in a document and $P(w_i|z_i = j)$ represent the probability of word, w_i, given topic j. Then the model specifies that probability of a word in a document as,

$$P(w_i) = \sum_{j=1}^{T} P(w_i|z_i = j)P(z_i = j), \tag{5.18}$$

where T is the number of topics. If we denote $\phi^j = P(w|z = j)$ and $\theta^d = P(z)$ as the probability distribution of topics for document d, then the parameters, ϕ and θ indicate which topics are important for which document and which words are important for which topic. The estimation of these two parameters is done using efficient estimation procedures like Gibbs Sampling.

Another method similar to topic modeling for extracting semantically related topics is Latent Semantic Analysis (LSA) (Deerwester et al., 1990). LSA uses a low-rank approximation, using Singular Value Decomposition (SVD), to the term-document matrix with minimal error according to the Frobenius norm. The topic model, can also be viewed as matrix factorization as pointed by (Hofmann, 1999b). In LDA, the word-document co-occurrence matrix matrix is split into two factors, a topic matrix ϕ and a document matrix θ. This suggests that both LSA and LDA perform a dimensionality reduction on the term-document co-occurrence matrix based on the term-term correlations.

In the same sense, our algorithm too performs dimensionality reduction. The difference is that our algorithm performs this dimensionality reduction in two stages. In the first stage, with the help of a background corpus and using the keyphrase detection module, we prune out most of the words that are not very relevant to the set of documents. In the second stage, we make use of the co-occurrence statistics between the n-grams to extract a set of distinct topical n-grams.

To strengthen our argument for using the keyphrase detection module as a dimensionality reduction step, we compare the performance of LDA using all the n-grams in the document and the performance of LDA using just the top k terms outputted by keyphrase detection module. In this experiment, we use the same number of n-grams as used for our algorithm, $k = 300$.

In evaluating the topic detection part, there exists two categories of methods, *intrinsic* and *extrinsic* (Liddy, 2001). Extrinsic methods evaluate the labels against a particular task, intrinsic methods measure the quality of the labels directly. We provide intrinsic and extrinsic evaluations of our algorithm.

To evaluate our facet detection algorithm, we created a gold standard list of facets for each data set. A list of the top 300 keyphrases generated by the KL divergence module was given to two evaluators. The evaluators labeled each keyphrase as a

Data set	**Coverage**	**Average pairwise JC**	**Normalized KL divergence**	**Evaluator rating**
iphone				
Gold standard	12,977	0.02	2.81	3.13
MMR-variant	10,321	0.04	2.45	2.20
Graph-based method	9,850	0.01	1.98	2.82
Adaptive Graph-based method	11,232	0.03	2.62	3.05
petfoodrecall				
Gold standard	2,659	0.05	4.30	3.43
MMR-variant	3,087	0.09	2.54	2.34
Graph-based method	2,055	0.01	1.75	2.81
Adaptive Graph-based method	2,972	0.04	3.42	3.12
spitzer				
Gold standard	4,036	0.03	2.29	3.31
MMR-variant	4,217	0.07	2.09	2.12
Graph-based method	2,468	0.01	1.60	2.88
Adaptive Graph-based method	4,108	0.05	2.12	3.22
vtech				
Gold standard	5,058	0.03	2.79	3.76
MMR-variant	8,536	0.11	2.45	2.27
Graph-based method	4,342	0.01	1.66	3.28
Adaptive Graph-based method	5,511	0.04	2.58	3.23

Table 5.2: Coverage, overlap and relevance and evaluation scores for the gold standard, baseline and graph-based method and the improved graph-based method

Data set	**Precision**	**Recall**	**F-score**
iphone			
MMR-variant	0.32	0.30	0.31
LDA	0.41	0.39	0.40
LDA using Keyphrase Detection	0.48	0.59	0.53
Graph-based method	0.52	0.60	0.56
Adaptive Graph-based method	0.66	0.58	0.62
petfoodrecall			
MMR-variant	0.42	0.49	0.45
LDA	0.50	0.62	0.55
LDA using Keyphrase Detection	0.59	0.68	0.63
Graph-based method	0.61	0.57	0.59
Adaptive Graph-based method	0.69	0.60	0.64
spitzer			
MMR-variant	0.47	0.48	0.47
LDA	0.59	0.65	0.62
LDA using Keyphrase Detection	0.67	0.70	0.68
Graph-based method	0.79	0.59	0.68
Adaptive Graph-based method	0.83	0.72	0.77
vtech			
MMR-variant	0.45	0.56	0.50
LDA	0.57	0.62	0.59
LDA using Keyphrase Detection	0.67	0.75	0.70
Graph-based method	0.72	0.76	0.74
Adaptive Graph-based method	0.82	0.74	0.78

Table 5.3: Precision, recall and F-score for the baseline, LDA, LDA using keyphrase detection,graph-based algorithm, and the Improved graph-based method

iPhone	**petfoodrecall**	**spitzer**	**vtech**
0.62	0.86	0.77	0.88

Table 5.4: Kappa scores for the gold standard

positive example of a topic or a negative example of a topic.

Cohen's Kappa coefficient(Cohen, 1960) was calculated for the gold standard. Table 5.4 lists the κ value for the four data sets.

The kappa scores for the petfoodrecall and vtech datasets showed good agreement among the raters, while the spitzer dataset had only fair agreement. For the iPhone data set, both evaluators had a large amount of disagreement on what they considered topics.

A separate group of evaluators was given the output from our graph-based algorithm, a list of the top KL divergence keyphrases of the same length, and the gold standard for all four datasets. Evaluators were asked to rate the keyphrases on a scale from one to five, with one indicating a poor topic, and five indicating a good topic. The number k of topics for the algorithm was cutoff where the f-score is maximized. The same number of phrases was chosen for KL divergence as well. Table 5.6 lists the cutoffs for the four datasets.

In addition, the precision, F-score, coverage and average pairwise Jaccard coefficient were calculated for the four data sets. Precision, recall and the F-score are given in table 5.3. The precision, recall and F-score for the gold standards is one. The others are shown in table 5.2. Average pairwise Jaccard coefficient is calculated by finding the Jaccard coefficient between every pair of topics in the output and averaging this value. This value is a measure of the redundancy. The average relevance

Spitzer	**Petfood recall**
Ashley Alexandra Dupre	Under Wal-Mart
Oberweis	Xuzhou Anying
Emperor's club	People who buy
Governor of New	Cuts and Gravy
Spitzer's resignation	Cat and Dog
Dr Laura	Cats and Dogs
Mayflower hotel	Food and Drug
Sex workers	Cyanuric acid
former New york	recent pet
High priced prostitution	industrial chemical
McGreevey	massive pet food
Geraldine Ferraro	Royal canin
High priced call	Iams and Eukanuba
legally blind	Dry food
money laundering	
Virginia Tech shooting	**iPhone**
Korean American	Photo sent from
Gun Ownership	Waiting in line
Holocaust survivor	About the iPhone
Mentally ill	Unlimited data
Shooting spree	From my iPhone
Don Imus	Cell Phones
Video Games	Multi-touch
Gun free zone	Guided tour
West Ambler Johnston	iPhone Launch
Columbine High school	Walt Mossberg
Self defense	Apple Inc
Two hours later	Windows Mobile
Gun violence	June 29th
Seung Hui Cho	Web Browser
Second Amendment	Activation
South Korean	

Table 5.5: Different topics chosen by the graph-based algorithm for the different datasets. Each data set contains a different number of documents, hence the number of topics chosen for each data set is diferent.

iphone	**petfoodrecall**	**spitzer**	**vtech**
25	30	24	18

Table 5.6: Number of generated topics for each collection.

2 Topics	**5 Topics**	**10 topics**	**20 topics**
Virginia Tech Shooting	Virginia Tech	Massacre at Virginia	Korean American
	Cho Seung	Gun Control	Columbine
	West Ambler Johnston	Norris Hall	Tragic
	Deadliest Shooting	West Ambler Johnston	Imus
	Tragic	Video Games	Video Games
		Gun Laws	Cho Seung
		Holocaust Survivor	West Ambler Johnston
		Cho Seung Hui	Norris Hall
		Korean American	Email
		Mental Health	Two hours later
			Massacre at Virginia
			High School
			Gun Laws
			Holocaust Survivor
			People
			Guns
			Media
			Gun Control
			Self Defense

Table 5.7: Different topics chosen by the adaptive algorithm for the Virginia Tech Tragedy data set

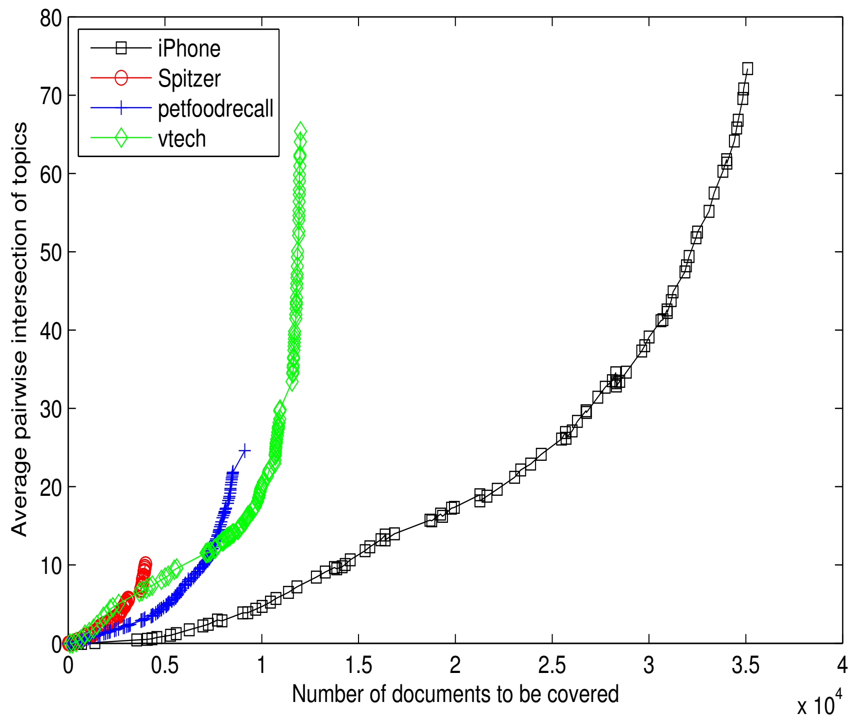


Figure 5.5: Subtopic redundancy vs. coverage. Each data set contains a different number of documents, hence the number of topics chosen for each data set is diferent.

is a normalized version of the combined “phraseness” and “informativeness” score calculated by the KL divergence method. This value is normalized by dividing by the KL divergence for the entire 300 phrase list. This provides a relevancy score for the ouput.

5.9 Results

Table 5.5 shows the different topics chosen by the graph-based algorithm for the different data sets. Our graph-based method performs very good and almost achieves the gold standard's rating. The F-score for the iPhone data set was only 0.56, but we believe part of this may be because this dataset did not have clearly defined topics, as shown by the low agreement (0.62) among human evaluators.

Table 5.7 shows the different topics chosen by the improved graph-based algorithm for the Virginia Tech Tragedy data set. It can be seen that when the number of topics requested is small, the chosen topics are representative yet diverse. The chosen topics are much more fine-grained as the number of topics increases.

Figure 5.5 shows the tradeoff between coverage and redundancy. This graph clearly shows that the overlap between the topics increases very slowly as compared to the number of documents covered. The slope of the curves increase slowly when the number of documents to be covered is small and later increases rapidly . This means that initially there are a lot of small focused topics and once we have selected all the focused ones the algorithm is forced to pick the bigger topics and hence the average pairwise intersection increases.

CHAPTER VI

Conclusion

6.1 Summary of Contributions

Similarity plays a very important role in many machine learning and natural language processing algorithms. We had three main goals in this thesis:

- Develop representation frameworks for similarity learning of relational data.
- Unsupervised similarity learning algorithms for relational data that uses the heterogeneous features
- Illustrate the usability of the learned similarity and representation framework in different applications.

This chapter summarizes our main contributions and describes future directions for research. In Chapter II, we describe different existing similarity measures for words, documents and structured objects like graphs and trees. Evaluating similarity measures is very hard since similarity measures are almost never used directly, but are rather used as a tool in many algorithms. Therefore, we describe both intrinsic and extrinsic evaluation approaches. In the intrinsic approach to evaluating similarity, similarity scores between selected pairs of objects are obtained from human experts. The correlation between the similarity scores outputted by the algorithm and the scores given by humans is used to evaluate similarity measures. If the similarity scores were obtained from more than one person, then the correlation between similarity scores by humans is used as an upper bound. In an extrinsic evaluation, the similarity scores are used in an external task and the performance in the task is used as the evaluation metric of the similarity measure. Some of the commonly used tasks for evaluating similarity measures are classification, clustering and recommendation systems. We describe the different algorithms for tasks which are used for evaluating similarity measures and the evaluation metrics for the tasks. To the best of our knowledge, there are no unsupervised learning algorithms which learn similarity using heterogeneous features which are present in relational data.

Chapters III and IV address the first two goals of the thesis. In Chapter III, we proposed a novel framework to represent objects with heterogeneous features which enables complex similarity learning. Assuming each object can be described by multiple features, we represent the objects using multiple graphs. Each graph is used to represent a single feature type. For example, in the case of publications, the publications form the vertex set of one graph while the associated keyphrases form the vertex set of another graph. Similarly, the associated authors, venues can be represented using multiple graphs. The edge weight in each graph represents the similarity between the different features. We formalized the problem of similarity estimation as an optimization problem induced by a regularization framework over edges in the multiple graphs. We show that there exists a direct solution to the convex optimization problem, albeit too expensive in terms of computational complexity. Therefore, we present an alternate algorithm for the optimization using the coordinate descent approach. We denote the proposed Similarity Measure using MUltiple Graphs as **SMUG**. We also illustrated the superior performance of SMUG in tasks like clustering and classication over baseline similarity measures: cosine similarity, citation similarity and linear combination of similarity measures. We also show that SMUG outperforms state of the art classication algorithm which uses spectral clustering of multiple graphs and co-clustering.

In Chapter IV, we extend the SMUG framework to allow real edge weights for edges across different layers of the graph. We explained the connection between similarity learning and feature weight learning. iteratively and simultaneously learn feature weights. We formalized the problem of similarity estimation as an optimization problem induced by the regularization framework over edges in multiple graphs. We developed an efficient, iterative algorithm based on Alternating Optimization (AO) which has a neat, intuitive interpretation in terms of random walks over multiple graphs.

We also presented theoretical equivalence results between node label regularization and edge weight regularization. We showed that all variants of SMUG can be easily parallelized under the MapReduce framework using a sequence of three MapReduces. We showed that the variant of SMUG which incorporates feature weight learning, in addition to, similarity learning outperforms SMUG, the baseline similarity measures and other standard similarity measures like Multiple Kernel Learning and Latent Semantic Analysis. We also illustrated the applicability of SMUG in a recommendation task using the Yahoo! KDD cup data set. In this problem, the task is to recommend different objects like tracks, genres, artists and albums to users based on past user preferences provided in the form of user ratings. We built a recommender system which incorporates user preferences and generalizes well using the available metadata. The recommender system accurately predicts user preference over genres, artists using the ratings for tracks. We compare the system against neighborhood models which do not differentiate between the different items (tracks, genres, artists and albums). The recommender system built on top of the SMUG framework outperforms models which do not exploit the available metadata.

In Chapter V, we develop adaptive algorithms which use the SMUG framework for topic detection in blogs. We present two new algorithms based on weighted set cover for finding topics in a corpus of blog posts. The algorithm uses the SMUG framework to represent documents and extracted candidate topic labels. We developed algorithms which adapt to the requested number of topics and selects representative and discriminative set of topics which can be used to describe the entire story. If the user selects a small number of topics, then the algorithm selects broad topics which convey the big picture of the story and if the user selects a large number of topics, the algorithm picks many fine-grained topics which shows the many different aspects/facets of the story Our algorithm provides a new method of ranking keyphrases that can help users find different facets of an event.

The identification of facets has many applications to natural language processing. Once facets have been identified in a collection, documents can be clustered based on these factets. These clusters can be used to generate document summaries or for visualization of the event space. The selected topics provide a succinct summary of the different topics.

We started with the problem of representing relational data for the purpose of learning similarity. We have shown that it is beneficial to represent objects using the different heterogeneous features and exploit the dependency between different features to learn similarity. We have also shown that multiple feature types can be used to learn similarity from each other and this further boosts the performance of the similarity measure.

6.2 Future Work

In this section, we outline some possible directions for future work. A possible avenue for future research is in applying active learning in the SMUG framework. (Raghavan et al., 2006; Druck et al., 2009) show that intelligently soliciting labels on multiple features facilitates more efficient annotation and leads to higher classification accuracy. We can combine the advantages of active learning and the availability of heterogeneous features to reduce annotation effort to obtain labeled data for tasks like classification. One possible idea is to represent the different feature similarity graphs as in Chapter III for active classification. A potential hypothesis is that we can exploit existing correlations between class labels of objects and features. For example, consider the problem of classifying scientific publications into three research areas:

Machine Translation, Dependeny Parsing and Summarization as in the AAN data set. Labeling the keyphrase "Statistical Machine Translation" as belonging to the class "Machine Translation" may be more helpful than labeling multiple publications if many publications contain the keyphrase "Statistical Machine Translation".

The key idea is to represent the different feature similarity graphs as in Chapter III for active classification. Essentially, once some of the object nodes are labeled, they impose soft-labels on the nodes in the other layers. The hypothesis is that if the features are useful in expressing the cluster structure of the object layer graph, then there exist consistent clusterings across the different layers. This can be best explained with help of an example. For example, consider the problem of classifying publications by research area. Let the features in consideration be authors, venues and keywords. Let the different research areas be "Machine Translation", "Summarization" and "Dependency Parsing", then once some publications are labeled as belonging to "Summarization" and written by "Dragomir Radev" or similar authors, then we can label another publication by "Dragomir Radev" as a summarization paper with high confidence. In other words, once some publications are labeled, it can be used to cluster other graph layers like the author layer. Once the different layers are classified using the initial set of labeled data, we can find how consistent are the different clusterings are. Using the consistency of different clusterings as the objective function, we can identify the set of nodes to be labeled so that other nodes can be labeled with high confidence while maintaining consistency of clusterings across different layers.

One of the concerns of the SMUG framework is the number of parameters (α_i) that need to be tuned. The number of parameters grows quadratically with the number of layers. In order to learn these parameters, we need training data in the form of class labels for a small fraction of the nodes. A possible extension to incorporate training data is to regularize node labels, in addition, to the edge weights. Thus, the modified algorithm can be used to regularize the node labels and edge weights. We can develop a new learning algorithm that can be used to tune the parameter values using the training data.

APPENDICES

APPENDIX A

Graph Clustering Using Active Learning

Introduction

Clustering is a very important problem with many applications in different areas. In general, clustering is an unsupervised task of grouping data points into groups (clusters). The research problem has been approached by researchers in many fields reflecting its broad appeal and usefulness in different areas. It is considered to be an ill-defined problem (Jain and Dubes, 1988) which in addition to its combinatorial nature makes it a very hard problem.

In many machine learning scenarios, there is a lot of unlabeled data whereas labeled data is very expensive. Active Learning is a learning technique which actively queries the user for labels. In general, the number of training samples required is much lower when active learning is used since the learning algorithm gets to pick the data which need to be labeled. In this paper, the main idea is to let the clustering algorithm pick a few data points to be labeled. Based on the labels provided by the user, the remaining points are clustered. The clustering algorithm tries to pick points which are mutually far away from each other and hence the set of points chosen form a representative set of points for the underlying community structure. As a preprocessing step, the algorithm augments the similarity matrix by propagating similarity values along edges. The only input to our algorithm is the similarity matrix and the cluster labels for the nodes selected by our algorithm.

The rest of the paper is organized as follows. Section 2 reviews the previous work in this area. In section 3, we describe the new algorithm and then we introduce our data set in section 4. Section 5 describes the experiments conducted and in section 6 we explain our evaluation process followed by a discussion of the results. We conclude this paper in section 8 with a brief summary of the performance and possible directions for future work.

Related Work

There has been a lot of previous work (MacQueen, 1967; Heyer et al., 1999; Shi and Malik, 2000) in Clustering. A good recent survey of the different approaches can be found in (Grira et al.). Semi-supervised learning is a class of machine learning algorithms that make use of both unlabeled and labeled data. They have been found to perform very well (Belkin et al., 2004; florina Balcan et al., 2005; Blum and Chawla, 2001) with a limited amount of labeled data and a large amount of unlabeled data. This makes semi-supervised learning algorithms fall in between unsupervised and supervised learning algorithms . Due to the success of semi-supervised approaches, there has been a lot of work in semi-supervised clustering (Bilenko et al., 2004; Basu et al., 2004b). In this setting, the supervision (training data) has been in the form of constraints. The constraints are given in the form of answers to questions of the following form, "should the i^{th} and the j^{th} object be in the same cluster or not?"

It has been shown (Davidson et al., 2006; Wagstaff, 2006) that a larger number of constraints does not necessarily yield a better performance in semi-supervised clustering, i.e, most of the work on semi-supervised clustering algorithms report only "average" behavior when different constraint sets are used while it is possible that a particular set of constraints decreases performance over a smaller constraint set. This issue has led to the scarcity of active learning approaches for semi-supervised clustering(Mallapragada et al., 2008; Basu et al., 2004a). Therefore, in all the active clustering algorithms till now, the algorithm sequentially asks questions of the form "should the i^{th} object and the j^{th} object be linked together or not?" Both active learning algorithms have two phases, the Explore phase and the Consolidate phase. The explore phase selects must-not link constraints using a farthest-first strategy. The explore phase continues until k points are chosen such that no two points should link together, where k is the number of clusters. The two algorithms vary in the Consolidate phase. (Basu et al., 2004a) choose points randomly such that they are not selected as part of the k points and queries them against each of the k points, until a must-link query is obtained. In other work, (Mallapragada et al., 2008) assigns each unselected point to the closest point among the k chosen points. According to the active learning principle, points which are farthest away from the chosen points are selected during the Consolidate phase for querying.

Active Learning Algorithm

Given is a graph $G = (V, E, W)$ where V represents the set of nodes and E represents the set of edges. The function W gives the weight of an edges between two nodes. The edge weights represent the similarity between the nodes. We can view the similarity between two nodes as the probability that the two nodes belong to the same cluster. Therefore, once we know the class of one node, we can label nodes adjacent to it which are connected by an edge with weight above a threshold.

Definition A.1. If two nodes a and b are connected by an edge with weight $w_{ab} \geq \epsilon$, where ϵ is the threshold, then we say a is covered by b and vice versa.

We can also label nodes which are more than one hop away as long as the product of the edge weights along the path is greater than the threshold. We define the augmented similarity between two nodes a and b as the maximum product of edge weights along the path between a and b.

$$S^{'} = max_{p \in path(a,b)} path\_product(p) \tag{A.1}$$

where, $path(a, b)$ is the set of all the paths between a and b. Let the path p be represented as $(a, v_1, \ldots, v_k, b)$ and let $path\_product(p)$ be defined as

$$path\_product(p) = w_{a,v_1} \times \prod_{i=1}^{k-1} w_{v_i,v_i+1} \times w_{v_k,b} \tag{A.2}$$

Thus we need to select a set of nodes such that every node in the graph can be labeled. There are two phases in our algorithm. In the first phase, we compute the covering relation between all nodes. To compute the covering relation, we need to compute the augmented similarity $S^{'}(a,b)$ for all pairs of vertices (a,b). We use the Floyd-Warshall algorithm on a transformed graph to compute the covering relation. Specifically, the graph is transformed by modifying the edge weights. Let the edge weight between two nodes, a and b, be $w_{a,b}$, then this edge weight is transformed as shown below,

$$w^{'}_{a,b} = -\log(w_{a,b}) \tag{A.3}$$

Then clearly,

$$S^{'}(a,b) = e^{-d(a,b)} \tag{A.4}$$

where $d(a,b)$ is the shortest path distance between a and b in the transformed graph. Now given the sets of nodes covered by every node, we find the minimum set cover. The minimum set cover selects the minimum set of nodes, which when labeled can help label all the nodes in the graph. In traditional Set Cover we are given a collection of sets S over a universe U, and the goal is to select a minimal subset of S such that the whole universe, U is covered. In the case of our problem, both the universe, U and S is the set of nodes in the graph V.

For example, consider a graph G with 7 nodes $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ and assume the covering relation is as shown in Table A.1. It can be seen that the covering relation is reflexive, i.e, if v_i covers v_j, then v_j covers v_i as well. A minimum set cover solution for this example is $C = \{v_1, v_4, v_5\}$, i.e, 3 is the size of the smallest set that can cover all vertices.

Unfortunately finding the minimal set cover is NP-hard (Garey and Johnson,

Vertex label	Vertices Covered
v_1	v_2, v_3
v_2	v_1, v_4
v_3	v_1, v_5
v_4	v_2
v_5	v_6, v_7, v_3
v_6	v_5
v_7	v_5

Table A.1: Example showing the covering relation for a toy graph

1990), so we must settle for an approximate solution. But fortunately there exist very good α-approximation algorithms for this problem (Cui, 2007). The greedy algorithm for set covering chooses sets iteratively; it chooses the set which contains the largest number of uncovered elements until all the elements are covered. It can be shown (Cui, 2007) that this algorithm achieves an approximation ratio of $H(s) = O(\log |s|)$, where s is the size of the largest set and $H(n)$ is the n^{th} harmonic number. Let A be the set of nodes selected for labeling and let $R(A)$ denote the number of nodes covered by A using a particular threshold. Now clearly the following theorem holds true:

Theorem A.2. *For all sets, $A \subseteq B \subseteq V$ and a node $v \in V - B$, it holds that*

$$R(A \cup v) - R(A) \geq R(B \cup v) - R(B) \tag{A.5}$$

Hence, the function R is submodular. Therefore, the greedy algorithm that picks the set that covers maximum number of remaining nodes at every iteration is near optimal after every iteration (Nemhauser et al., 1978).

Let the set of nodes selected by our algorithm be defined as S. Then, once the selected set of nodes are labeled by the user, the remaining nodes are labeled using the following equation:

$$label(v_i) = label(\underset{v \in V - S_k}{\arg\max} S^{'}(v, v_i)) \tag{A.6}$$

Essentially, all the unlabeled points are assigned the cluster label of the closest point as given by the user. Unfortunatel, this algorithm has a time complexity of $O(|V|^3)$ which is prohibitively high for most real networks. Hence instead of using Floyd-Warshall's algorithm to compute the covering relation between vertices, we use a simple breadth first search to find out all vertices which are located within an augmented similarity score of ϵ. Since the product of the similarity values falls very quickly with the number of hops, this algorithm runs much faster empirically than the $O(|V|^3)$ algorithm using Floyd-Warshall.

In the ideal case, the algorithm should select exactly k nodes where k is the number of clusters in the graph. This will happen if the similarity function is an ideal similarity function. An ideal similarity function gives a score of 1 for any two nodes which should be in the same cluster and 0 otherwise. Clearly, we can see that when we have an ideal similarity function, the algorithm behaves ideally by selecting just k nodes for labeling.

In general, the clustering algorithm will perform well as long as the similarity function satisfies a very intuitive and basic property. The similarity function should give a high similarity score for inter cluster pairs as compared to intra cluster pairs. The amount of training data required depends on the gap between the similarity values between inter cluster pairs and intra cluster pairs. The bigger the gap between the similarity values of inter cluster pairs and intra cluster pairs, the more nodes are covered by each node in its own cluster. Hence, the number of nodes selected to be labeled goes down as the gap between the similarity value of inter cluster and intra cluster pairs increases.

To illustrate the working of our algorithm (Alg. A.1), consider the toy graph shown in Figure A.2. The edges are shown with varying width and darkness depending on the weight of the edge. The larger the edge weight, the thicker and darker the edge. Intuitively, there are three clusters in the graph. Vertices $(1, 2, 3, 4)$ form one cluster, $(5, 6, 7)$ form another cluster and the remaining three vertices form the third cluster.

The augmented graph when we use a threshold value of 0.3 is shown in A.3. The larger vertices are the vertices chosen for labeling. The edge weight between vertices,8 and 10, has been incremented to 0.576 because of the path $(8- > 9- > 10)$. Clearly the cluster structure becomes more prominent after the augmentation of similarities.

Now we apply minimum set cover on the augmented graph and vertex 3 is chosen first since it covers the maximum number of vertices. Then vertices 9 and 6 are chosen successively to cover the remaining vertices. Therefore $(3, 9, 6)$ is the set of chosen vertices for labeling. Once the user labels the vertices with different class labels, the remaining vertices are labeled using the closest labeled vertex.

```
Input: Graph $G = (V, E, W)$, threshold $\epsilon$.
Output: Set of nodes selected for labeling.
$N = |V|$
Augment graph G using Equation A.1
Let $W^{'}$ be the modified similarity function.
for $i = 1$ to $N$ do
   for $j = 1$ to $N$ do
      if ${w^{'}}_{v_j v_i} > \epsilon$ then
         $cover(v_i) = cover(v_i) \cup v_j$
      end if
   end for
end for
$nodes\_remaining = \{v_1, v_2, \ldots, v_N\}$
$selected\_nodes = \emptyset$
while nodes_remaining $\neq \emptyset$ do
   $v_{selected} = \arg\max_u |cover(u)|$
   $selected\_nodes = selected\_nodes \cup v_{selected}$
   $cover(v_i) = cover(v_i) - cover(v_{selected}) \forall i \in [1, n]$
end while
return $selected\_nodes$
```

Figure A.1: Active Learning Algorithm for Clustering Using Minimum Set Cover

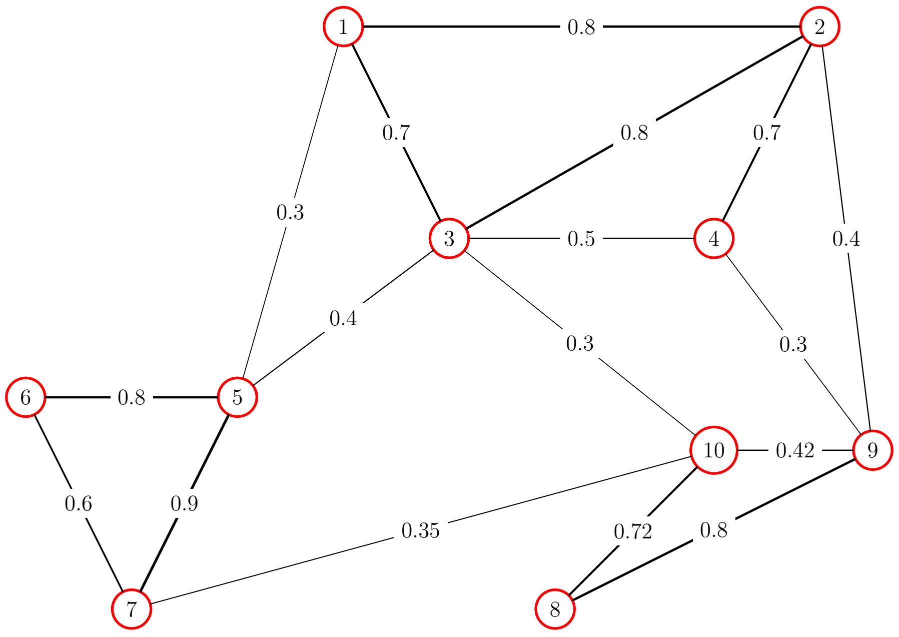


Figure A.2: Toy Graph

Data

We performed experiments on a wide variety of data sets. Below is the list of data sets.

- AAN: We use the classification data set derived from AAN (Radev et al., 2009b) described in Chapter III.

- We use three binary classification tasks derived from the 20 newsgroups data. The three binary classification tasks are, rec.sport.basketball (994 documents) vs. rec.sport.hockey (999 documents); comp.sys.ibm.pc.hardware (982 documents) vs. comp.sys.mac.hardware (961 documents); talk.religion.misc (628 documents) vs. alt.atheism (799 documents). Two documents u, v are con-

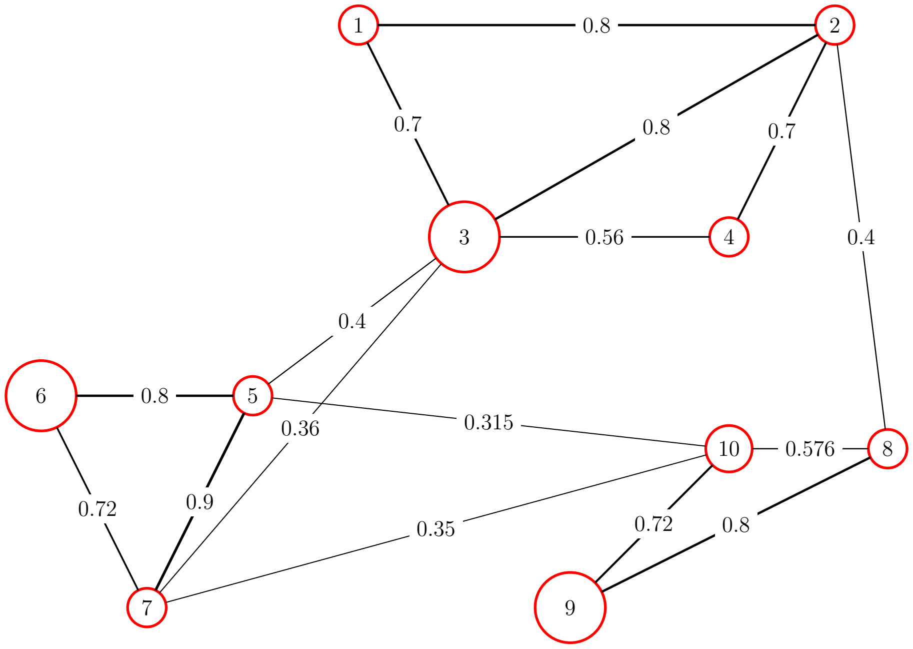


Figure A.3: Modified Graph

nected if u is among the 10 nearest neighbors of v or v is among the 10 nearest neighbors of u as measured by the cosine similarity (CS). We computed the cosine similarity between the documents after converting each document into a "tf.idf" vector without any preprocessing like header removal, stopword removal, stemming, frequency cutoff. We applied the following weight function on the edges $w_{uv} = e^{(CS_{uv}-1)/0.03}$ before computing the nearest neighbor graph.

- An interaction network between 216 yeast genes, where each gene is labeled with one of three KEGG (Ogata et al., 1999) functional pathway labels. This data is a subgraph of a high-quality probabilistic functional network of yeast genes (Lee et al., 2004). Each edge weight in this network represents a probability of linkage between two genes, estimated by integrating diverse functional genomic data sources.

- The handwritten digits data set originates from the Cedar Buffalo digits database (Hull, 1994). The digits were initially preprocessed to reduce the size of each image down to a 16×16 grid by down-sampling and Gaussian smoothing, with pixel values in 0 to 255 (LeCun et al., 1990). The standard deviation of the Euclidean distance between the images is 380. The edge weights between the images x_i and x_j is calculated as,

$$w_{x_i x_j} = e^{(-\sum_{d=1}^{256}(x_{i,d}-x_{j,d})^2/380^2)} \tag{A.7}$$

Experiments

We compare our "Greedy Algorithm" against the following algorithms

- *Zhu*: (Zhu et al., 2003a) algorithm employs active learning algorithm on top of semi-supervised learning. The algorithm uses a simple Bayes Classifier and a harmonic function to ensure label smoothness over the nodes in the graph. They define the true risk of the bayes classifier based on the harmonic function h as,

$$R(h) = \sum_{i=1}^{n} \sum_{y_i=0,1} [sgn(h_i) \neq y_i] p^*(y_i) \tag{A.8}$$

where $sgn(h_i)$ is the Bayes decision rule with threshold 0.5, such that $sgn(h_i) = 1$ if $h_i > 0.5$ and $sgn(h_i) = 0$ otherwise and y_i is the true label of the $i^t h$ node.

Here $p^*(yi)$ is the unknown true label distribution at node i, given the labeled data. $p^*(y_i = 1)$ is approximated as the probability of reaching 1 in a random walk on the graph. For performing the active learning, they greedily choose the point which will minimize the expected estimated error as the next point to be labeled. The user provides the true cluster label for the chosen point and they continue the process iteratively.

- *Random*: The Random baseline uses the graph updated with the augmented similarity as described in Section 3. Instead of using the minimum set cover to compute the set of nodes to be labeled, we pick nodes randomly to be labeled.

Results

Given that we have the cluster labels for all the nodes, we compare the quality of the clusterings obtained by the different algorithms using classification accuracy on the unlabeled data when different number of nodes are labeled. Classification accuracy is defined as the percentage of total number of nodes that are correctly labeled. Figure A.4 shows the classification accuracy obtained on the different data sets by all the algorithms when different number of points are labeled. It can be seen that our algorithm outperforms both Zhu et al's algorithm and the random baseline on all the data sets. On all the data sets, only a handful of data points need to be labeled to obtain more than 80% accuracy. The set of points picked by our algorithm to be labeled depends on the distribution of the cluster sizes, i.e, if a particular cluster is bigger, then there are more points picked to be labeled in that particular cluster.

Discussion

We compared our algorithm against the semi-supervised algorithm by (Kulis et al., 2009) too. Since the supervision for semi-supervised algorithms is not in the form of cluster labels for specific nodes, we cannot use accuracy as the metric of evaluation. Instead, we use Normalized Mutual Information(NMI) (Strehl and Ghosh, 2002) as our evaluation metric for comparison with semi-supervised algorithms. However, the semi-supervised algorithm achieved poor performance on all the data sets except the Gene Network. For example, the semi-supervised algorithm achieved a NMI score of 0.001 on the Hockey vs. Baseball data set with 500 constraints. We show that our setting and the semi-supervised algorithm's setting are equivalent in the following theorem which helps us compare algorithms in the active learning setting with semi-supervised clustering algorithms.

Theorem A.3. *Labeling n nodes in a graph G with k clusters is equivalent to answering a worst-case of $O(nk)$ pairwise constraints, where the constraints are answers to questions of the form "Should vertices u and v be in the same cluster or in different clusters?".*

PROOF: Given n nodes to be labeled in a graph G with k clusters, let the first node be labeled as C_1. The second node can be compared against the first node and if they should be in the same cluster, then label the second node as C_1, else as C_2. Let the second node be labeled as C_2. Now, the third node can be compared against both the first node and the second node and can be labeled as C_1 or C_2 or C_3 depending on the comparisons. Thus, it can be seen that labeling the i^{th} node requires answering

$max(i-1, k-1)$ queries. Therefore, in the worst case, it takes $O(nk)$ queries to be answered to label n nodes in a graph G with k clusters.

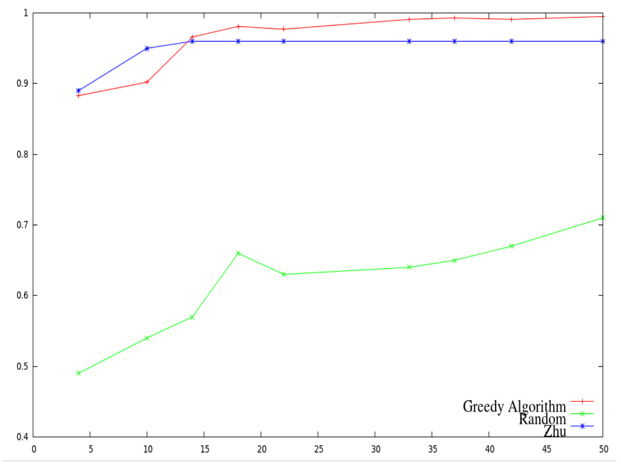
Therefore, for the Hockey vs. Baseball data set answering 500 queries is equivalent to labeling 250 nodes with cluster labels. Our algorithm achieves the maximum NMI score of 1.0 when 250 nodes are labeled. The semi-supervised algorithm does not perform well because the data set contains a total of $\frac{1993*1992}{2} = 1,985,028$ edges and labeling 500 of the edges is very little supervision compared to labeling 250 nodes.

Querying nodes iteratively for the cluster label is not conducive for active learning for clustering since it requires the user to have a broad overview of the entire data and the underlying clusters. To illustrate this clearly, consider the following scenario. Suppose, we are trying to cluster publications in two fields, Medicine and Engineering. Let the Medicine cluster consist of two subclusters, Cardiology and Ophthalmology. Let the Engineering cluster consist of two subclusters, Electrical Engineering and Computer Science. Now if the first query posed by the active learning algorithm is "Should an Electrical Engineering publication and a Computer Science publication be linked together?", the user has no idea about how it should be answered because the user does not know the resolution of the clustering he/she needs. Therefore, for active learning in this setting to be useful, the user needs to get a broad overview of the entire data set and the underlying community structure. On the contrary, our algorithm would select a set of representative points from all the clusters (Electrical Engineering, Computer Science, Cardiology and Ophthalmology) for being labeled by the user. This provides the user with a compressed snapshot of the entire data set and the user can choose the resolution of clustering he/she needs. Hence our algorithm helps visualize a set of points by reducing the total number of points to a few points representative of each cluster. Therefore, it is better to ask the cluster labels of a set of points at once instead of asking the queries sequentially which requires the user to know the underlying clusters.

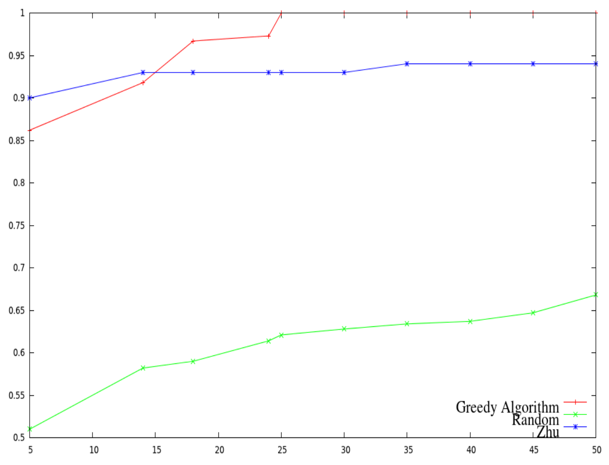
Conclusions and Future Work

In this work, we developed an active graph clustering algorithm. We formulated the clustering problem as an optimization problem and proposed a greedy algorithm with strong performance guarantees. We have shown that since the objective we are optimizing is submodular, the greedy algorithm achieves an optimal solution within a constant factor.

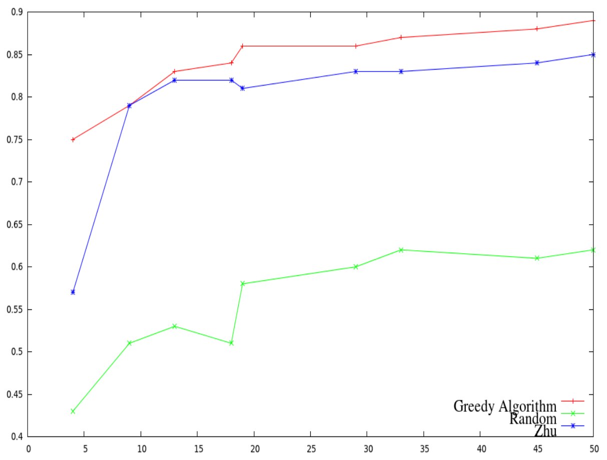
We evaluated our algorithm on many different datasets and our algorithm outperforms baseline algorithms and a state of the art clustering algorithm.



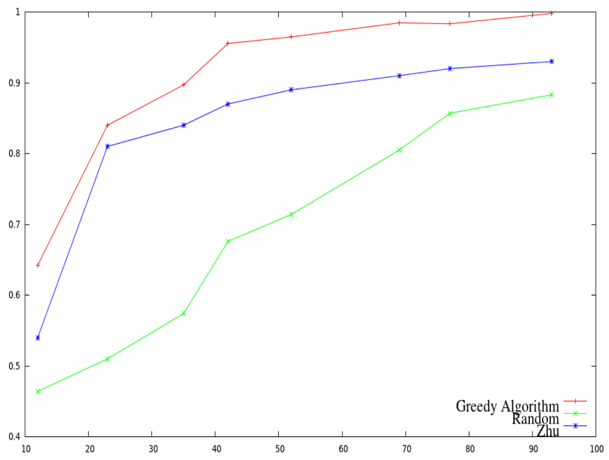
(a) Baseball vs Hockey



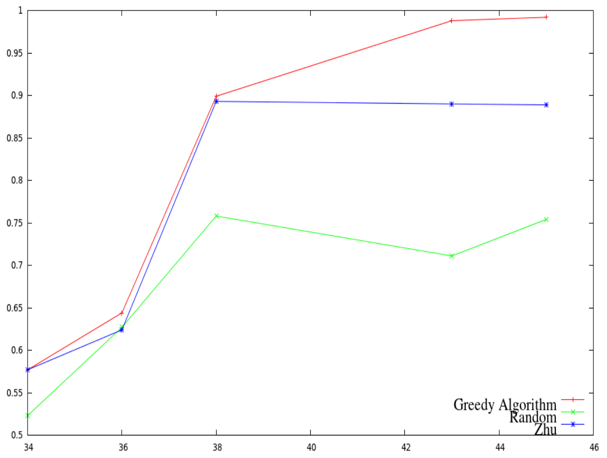
(b) PC vs MAC



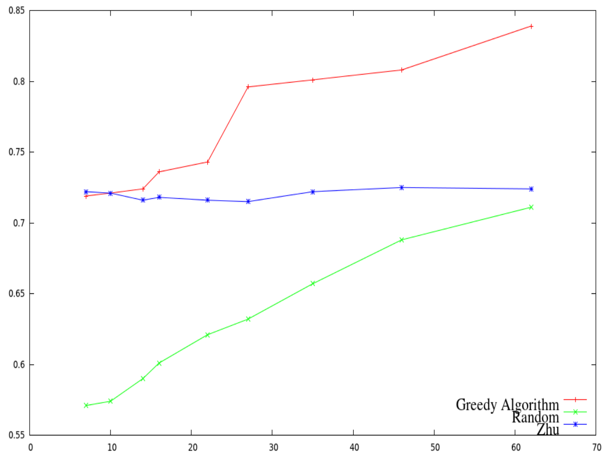
(c) Religion vs Atheism



(d) OCR 10 Digits



(e) Gene Network



(f) AAN

Figure A.4: Classification Accuracy on the different data sets. The number of points labeled is plotted along the x-axis and the y-axis shows the classification accuracy on the unlabeled data.

APPENDIX B

ACL Anthology Network (AAN) Corpus

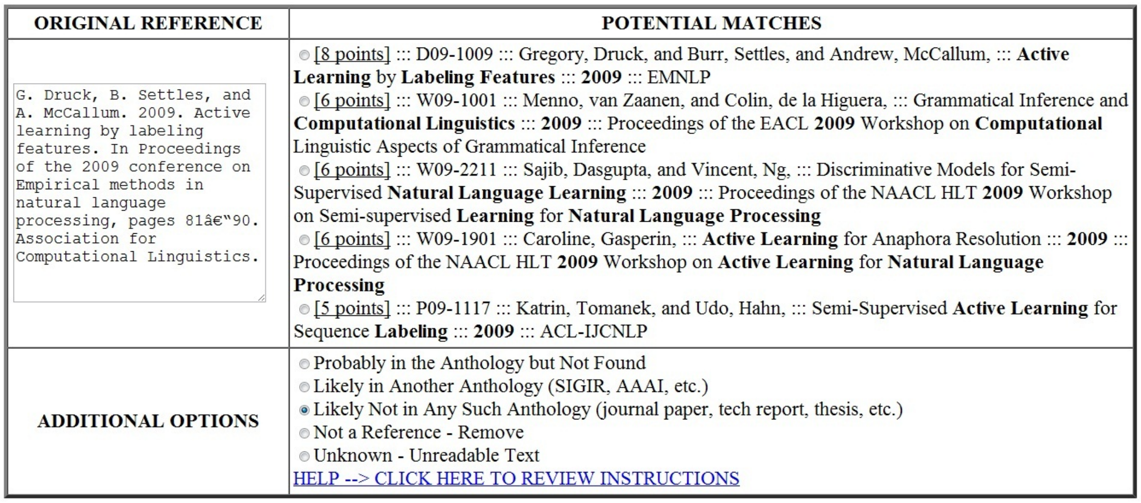
Introduction

The ACL Anthology is one of the most successful initiatives of the ACL. It was initiated by Steven Bird and is now maintained by Min Yen Kan. It includes all papers published by ACL and related organizations as well as the Computational Linguistics journal over a period of four decades. It is available at http://www.aclweb.org/anthology-new/ . One fundamental problem with the ACL Anthology, however, is the fact that it is just a collection of papers. It doesnt include any citation information or any statistics about the productivity of the various researchers who contributed papers to it. We embarked on an ambitious initiative to manually annotate the entire Anthology in order to make it possible to compute such statistics. In addition, we were able to use the annotated data for extracting citation summaries of all papers in the collection and we also annotated each paper by the gender of the authors (and are currently in the process of doing similarly for their institutions) in the goal of creating multiple gold standard data sets for training automated systems for performing such tasks.

Curation

The ACL Anthology includes 17,610 papers (excluding book reviews). We converted each of the papers from pdf to text using an OCR tool (www.pdfbox.org). After this conversion, we extracted the references semi-automatically using string matching. The above process outputs all the references as a single block of continuous running text without any delimiters between references. Therefore, we manually inserted line breaks between references. These references were then manually matched to other papers in the ACL Anthology using a "k-best" (with k = 5) string matching algorithm built into a CGI interface. A snapshot of this interface is shown in Figure B.1. The matched references were stored together to produce the citation network. In the case of the citation not being found, we have 5 different options the user can choose from. The first option is Possibly in the anthology but not found which is used if the string similarity measure failed to match the citation to the paper in AAN. The second option, Likely in another anthology is used if the citation is for a paper in a related conference. We considered the following conferences as related conferences AAAI, AMIA, ECAI, IWCS, TREC, ECML, ICML, NIPS, IJCAI, ICASSP, ECIR, SIGCHI, ICWSM, EUROSPEECH, MT, TMI, CIKM and WWW.

The third option is used if the cited paper is a journal paper or a technical report or PhD thesis or a book. The last two options are used if the reference is not readable because of an error in the PDF to text conversion or if it is not a reference. Only references to papers within AAN are used in the computation of statistics. In order to fix the issue of wrong author names and multiple author identities we had to perform a lot of manual post-processing. The first names and the last names were swapped for a lot of authors. For example, the author name “Caroline Brun” was present as “Brun Caroline” in some of her papers. Another big source of error was the exclusion of middle names or initials in a number of papers. For example, Julia Hirschberg had two identities as “Julia Hirschberg” and “Julia B. Hirschberg”. There were many spelling mistakes, like “Madeleine Bates” was misspelled as “Medeleine Bates”. There were about 1000 such errors that we had to manually correct. In some cases, the wrong author name was included in the metadata and we had to manually prune such author names. For example, “Sofia Bulgaria”, “Thomas J. Watson” was incorrectly included as author names. Also, there were cases of duplicate papers being included in the anthology. For example, C90-3090 and C90-3091 are duplicate papers and we had to remove such papers. Finally, many papers included incorrect titles in their citation sections. Some used the wrong years and/or venues as well. For example, here is one reference to a paper with the wrong venue. “Hiroshi Kanayama Tetsuya Nasukawa. 2006. Fully Automatic Lexicon Expansion for Domain-oriented Sentiment Analysis. In ACL.” The cited paper itself was published in EMNLP 2006 and not ACL 2006 as shown in the reference. In some cases, the wrong conference name was included in the metadata itself. For example, W07-2202 had “IJCNLP” as the conference name in the metadata while the right conference name is “ACL”. Conference names were normalized to canonical names. For example, joint conferences like “COLING-ACL” had "ACL-COLING" as the conference name for some papers and "COLING-ACL" in some other papers.

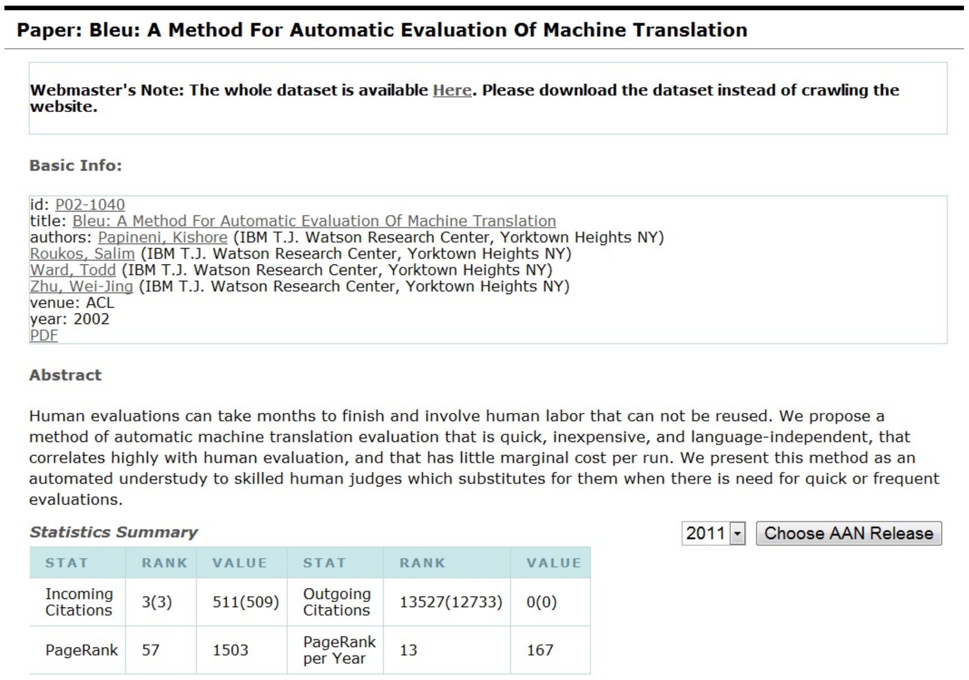


ORIGINAL REFERENCE	POTENTIAL MATCHES
G. Druck, B. Settles, and A. McCallum. 2009. Active learning by labeling features. In Proceedings of the 2009 conference on Empirical methods in natural language processing, pages 81â€"90. Association for Computational Linguistics.	○ [8 points] ::: D09-1009 ::: Gregory, Druck, and Burr, Settles, and Andrew, McCallum, ::: **Active Learning** by **Labeling Features** ::: **2009** ::: EMNLP ○ [6 points] ::: W09-1001 ::: Menno, van Zaanen, and Colin, de la Higuera, ::: Grammatical Inference and **Computational Linguistics** ::: **2009** ::: Proceedings of the EACL **2009** Workshop on **Computational** Linguistic Aspects of Grammatical Inference ○ [6 points] ::: W09-2211 ::: Sajib, Dasgupta, and Vincent, Ng, ::: Discriminative Models for Semi-Supervised **Natural Language Learning** ::: **2009** ::: Proceedings of the NAACL HLT **2009** Workshop on Semi-supervised **Learning** for **Natural Language Processing** ○ [6 points] ::: W09-1901 ::: Caroline, Gasperin, ::: **Active Learning** for Anaphora Resolution ::: **2009** ::: Proceedings of the NAACL HLT **2009** Workshop on **Active Learning** for **Natural Language Processing** ○ [5 points] ::: P09-1117 ::: Katrin, Tomanek, and Udo, Hahn, ::: Semi-Supervised **Active Learning** for Sequence **Labeling** ::: **2009** ::: ACL-IJCNLP
ADDITIONAL OPTIONS	○ Probably in the Anthology but Not Found ○ Likely in Another Anthology (SIGIR, AAAI, etc.) [✓ Likely Not in Any Such Anthology (journal paper, tech report, thesis, etc.)] ○ Not a Reference - Remove ○ Unknown - Unreadable Text HELP --> CLICK HERE TO REVIEW INSTRUCTIONS

Figure B.1: CGI interface used for matching new references to existing papers

Basic Statistics

Using the metadata and the citations extracted after curation, we have built three different networks. The paper citation network is a directed network with each node representing a paper labeled with an ACL ID number and the edges representing a citation within that paper to another paper represented by an ACL ID. The paper citation network consists of 17,610 papers and 77,048 citations. The author citation network and the author collaboration network are additional networks derived from the paper citation network. In both of these networks a node is created for each unique author. In the author citation network an edge is an occurrence of an author citing


Paper: Bleu: A Method For Automatic Evaluation Of Machine Translation

Webmaster's Note: The whole dataset is available Here. Please download the dataset instead of crawling the website.

Basic Info:

id: P02-1040
title: Bleu: A Method For Automatic Evaluation Of Machine Translation
authors: Papineni, Kishore (IBM T.J. Watson Research Center, Yorktown Heights NY)
Roukos, Salim (IBM T.J. Watson Research Center, Yorktown Heights NY)
Ward, Todd (IBM T.J. Watson Research Center, Yorktown Heights NY)
Zhu, Wei-Jing (IBM T.J. Watson Research Center, Yorktown Heights NY)
venue: ACL
year: 2002
PDF

Abstract

Human evaluations can take months to finish and involve human labor that can not be reused. We propose a method of automatic machine translation evaluation that is quick, inexpensive, and language-independent, that correlates highly with human evaluation, and that has little marginal cost per run. We present this method as an automated understudy to skilled human judges which substitutes for them when there is need for quick or frequent evaluations.

Statistics Summary

STAT	RANK	VALUE	STAT	RANK	VALUE
Incoming Citations	3(3)	511(509)	Outgoing Citations	13527(12733)	0(0)
PageRank	57	1503	PageRank per Year	13	167

Figure B.2: Snapshot of the different statistics for a paper

Year	Network			
		Paper Citation Network	Author Citation Network	Author Collaboration Network
2006	n	8898	7849	7849
	m	38,765	137,007	41,362
2007	n	9767	9421	9421
	m	44,142	158,479	45,878
2008	n	13,706	11,337	11,337
	m	54,538	196,505	57,614
2009	n	14,912	12,499	12,499
	m	61,527	230,658	63,772
2010	n	16,962	13,398	13,398
	m	72,463	278,366	73,044
2011	n	17,610	13,692	13,692
	m	77,048	297,924	76,206

Table B.1: Growth of Citation Volume

another author. For example, if a paper written by Franz Josef Och cites a paper written by Joshua Goodman, then an edge is created between Franz Josef Och and Joshua Goodman. Self citations cause self loops in the author citation network. The author citation network consists of 13,692 unique authors and 515,593 edges. There exist many duplicate edges, that is, the same author citing another author. The author citation network consists of 297,924 edges if duplicates are removed. In the author collaboration network, an edge is created for each collaboration. For example, if a paper is written by Franz Josef Och and Hermann Ney, then an edge is created between the two authors. Table B.1 shows some brief statistics about the different releases of the data set (2006, 2007, 2008, 2009. 2010 and 2011). Table B.2 shows statistics about the number of papers in some of the renowned conferences in Natural Language Processing. Figure B.2 shows statistics for an example paper.

Venue	**Number of Papers**
COLING	3482
ACL	2653
Computational Linguistics	900
EACL	782
EMNLP	1172
CoNLL	482
ANLP	334

Table B.2: Statistics for popular venues

Text Classification

We created a relational data set for text classification. We chose a subset of papers in 3 topics (Machine Translation, Dependency Parsing, and Summarization) from the ACL anthology. These topics are three major research areas in Natural Language Processing (NLP). Specifically, we collected all papers which were cited by papers whose titles contain any of the following phrases, "Dependency Parsing", "Machine Translation", "Summarization". From this list, we removed all the papers which contained any of the above phrases in their title because this would make the classification task easy. The pruned list contains 1190 papers. We manually classified each paper into four classes (Dependency Parsing, Machine Translation, Summarization, Other) by considering the full text of the paper. The manually cleaned data set consists of 275 Machine Translation papers, 73 Dependency Parsing papers and 32 Summarization papers for a total of 380 papers.

This data set is different from other text classification data sets in the sense that there are many relational features that are provided for each paper, like textual information, citation information, authorship information, venue information. Recently, There has been a lot of interest in computing better similarity measures for objects by using all the features ”together” [Zhou et al., 2008]. Since it is very hard to evaluate similarity measures directly, they are evaluated extrinsically using a task for which a good similarity measure directly yields better performance, such as classification.

Downloads

The text files, metadata of all the publications can be downloaded. In addition to that, the paper citation network, author citation network and collaboration network are available for download. Figure B.3 shows a snippet of the data available for download.

We also include a large set of scripts which use the paper citation network and the metadata file to output the auxiliary networks and the different statistics. The scripts are documented here: http://clair.si.umich.edu/anthology/ .The data set has already been downloaded from 6930 unique IPs since June 2007. Also, the website has been very popular based on access statistics. There have been nearly 1.1M hits between April 1, 2009 and March 1, 2010. Most of the hits were searches for papers or authors.

```
id      = {C98-1096}
author  = {Jing, Hongyan; McKeown, Kathleen R.}
title   = {Combining Multiple, Large-Scale Resources in a Reusable
   Lexicon for Natural Language Generation}
venue   = {International Conference On Computational Linguistics}
year    = {1998}

id      = {J82-3004}
author  = {Church, Kenneth Ward; Patil, Ramesh}
title   = {Coping With Syntactic Ambiguity Or How To Put The Block
   In  The Box On The Table}
venue   = {American Journal Of Computational Linguistics}
year    = {1982}

A00-1001 ==> J82-3002
A00-1002 ==> C90-3057
C08-1001 ==> N06-1007
C08-1001 ==> N06-1008
```

Figure B.3: Sample contents of the downloadable corpus

BIBLIOGRAPHY

BIBLIOGRAPHY

Abney, S. (2002), Bootstrapping, in ACL, pp. 360–367, Association for Computational Linguistics, Morristown, NJ, USA.

Ahlgren, P., B. Jarneving, and R. Rousseau (2003), Requirements for a cocitation similarity measure, with special reference to Pearson's correlation coefficient, JASIS, 54(6), 550–560.

Ahlgren, P., B. Jarneving, and R. Rousseaul (2004), Author cocitation analysis and pearson's correlation coefficient, JASIS, 55(9), 1250–1259.

Albert, R. Z. (2001), Statistical mechanics of complex networks, Ph.D. thesis, Notre Dame, IN, USA, aAI3000268.

Allan, J. (2002a), Introduction to topic detection and tracking, pp. 1–16, Kluwer Academic Publishers.

Allan, J. (2002b), Topic Detection and Tracking: Event-Based Information Organization, Kluwer Academic Publishers.

Allan, J., C. Wade, and A. Bolivar (2003), Retrieval and novelty detection at the sentence level, in ACM SIGIR, pp. 314–321, ACM, Toronto, Canada.

Allan, J., et al. (1998), Topic detection and tracking pilot study final report, in In Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop, pp. 194–218.

Amin, M., and M. A. Mabe (2003), Impact factors: use and abuse., Medicina, 63(4), 347–354.

Bach, F. R., G. R. G. Lanckriet, and M. I. Jordan (2004), Multiple kernel learning, conic duality, and the smo algorithm, in ICML, pp. 6–15, ACM, New York, NY, USA.

Backstrom, L., D. Huttenlocher, J. Kleinberg, and X. Lan (2006), Group formation in large social networks: Membership, growth, and evolution, in KDD, pp. 44–54.

Balcan, M. F., and A. Blum (2005), A pac-style model for learning from labeled and unlabeled data, in COLT, pp. 111–126.

Bansal, N., and N. Koudas (2007), Blogscope: spatio-temporal analysis of the blogosphere, in WWW, pp. 1269–1270, ACM, Banff, Alberta, Canada.

Basu, S., A. Banerjee, and R. J. Mooney (2004a), Active semi-supervision for pairwise constrained clustering, in SDM, pp. 333–344.

Basu, S., M. Bilenko, , R. J. Mooney, and R. J. Mooney (2004b), A probabilistic framework for semi-supervised clustering, in KDD.

Belew, R. K. (2005), Scientific impact quantity and quality: Analysis of two sources of bibliographic data, Tech. Rep. cs.IR/0504036, Arxiv.org.

Belkin, M., I. Matveeva, and P. Niyogi (2004), Regularization and semi-supervised learning on large graphs, LNCS, 3120, 624–638.

Beygelzimer, A., S. Kakade, and J. Langford (2006), Cover trees for nearest neighbor, in ICML, pp. 97–104.

Bezdek, J., and R. Hathaway (2002), Some notes on alternating optimization, in Advances in Soft Computing (AFSS) 2002, LNCS, vol. 2275, edited by N. Pal and M. Sugeno, pp. 187–195, Springer Berlin.

Bilenko, M., S. Basu, and R. J. Mooney (2004), Integrating constraints and metric learning in semi-supervised clustering, in ICML, pp. 123–133, ACM, New York, NY, USA.

Bird, S., et al. (2008), The ACL anthology reference corpus: A reference dataset for bibliographic research in computational linguistics, in LREC.

Blei, D. M., A. Y. Ng, and M. I. Jordan (2003), Latent dirichlet allocation, Journal of Machine Learning Research, 3, 993–1022.

Bloehdorn, S., and R. Moschitti (2007), Combined syntactic and semantic kernels for text classification, in ECIR, vol. 4425.

Blum, A., and S. Chawla (2001), Learning from labeled and unlabeled data using graph mincuts, in ICML.

Blum, A., and T. Mitchell (1998), Combining labeled and unlabeled data with co-training, in COLT, pp. 92–100, ACM, New York, NY, USA.

Bollen, J., H. V. den Sompel, J. A. Smith, and R. Luce (2005), Toward alternative metrics of journal impact: A comparison of download and citation data, IPM, 41(6), 1419–1440.

Bollen, J., M. A. Rodriguez, and H. Van de Sompel (2006), Journal status, Scientometrics, 69, 669.

Bonacich, P. (1987), Power and centrality: A family of measures, AJS, 92(5), 1170–1182.

Börner, K., C. Chen, and K. W. Boyack (2003), Visualizing knowledge domains, Annual Review of Information Science and Technology, 37.

Boyd, S., and L. Vandenberghe (2004), Convex Optimization, Cambridge University Press.

Brooks, C. H., and N. Montanez (2006), Improved annotation of the blogosphere via autotagging and hierarchical clustering, in WWW, pp. 625–632, ACM, Edinburgh, Scotland.

Budanitsky, A. (2001), Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures, in NAACL Workshop on WordNet and other Lexical Resources.

Budanitsky, A., and G. Hirst (2006), Evaluating wordnet-based measures of lexical semantic relatedness, Computational Linguistics, 32(1), 13–47.

Bunescu, R., and R. J. Mooney (2005), Subsequence kernels for relation extraction, in NIPS, pp. 171–178.

Buyukkokten, O., H. Garcia-Molina, and A. Paepcke (2001), Seeing the whole in parts: text summarization for web browsing on handheld devices, in WWW, pp. 652–662, ACM, Hong Kong, Hong Kong.

Calado, P., M. Cristo, E. Moura, N. Ziviani, B. Ribeiro-Neto, and M. A. Gonçalves (2003), Combining link-based and content-based methods for web document classification, in CIKM, pp. 394–401, ACM, New York, NY, USA.

Carbonell, J. G., and J. Goldstein (1998), The use of MMR, diversity-based reranking for reordering documents and producing summaries, Research and Development in Information Retrieval, pp. 335–336.

Censor, Y. A., and S. A. Zenios (1997), Parallel Optimization: Theory, Algorithms and Applications, Oxford University Press.

Charikar, M. S. (2002), Similarity estimation techniques from rounding algorithms, in STOC, pp. 380–388, ACM.

Charniak, E. (1999), A maximum-entropy-inspired parser, in ANLP, pp. 132–139.

Chi, Y., B. L. Tseng, and J. Tatemura (2006), Eigen-trend: trend analysis in the blogosphere based on singular value decompositions, CIKM, pp. 68–77.

Christopherson, M. (2004), Identifying core documents with a multiple evidence relevance filter, Scientometrics, 61(3), 385–394.

Chvatal, V. (1979), A greedy heuristic for the set-covering problem, Mathematics of Operations Research, 4, 233–235.

Clauset, A., M. E. J. Newman, , and C. Moore (2004), Finding community structure in very large networks, Physical Review E, pp. 1– 6, doi:10.1103/PhysRevE.70.066111.

Cohen, J. (1960), A coefficient of agreement for nominal scales, Educational and Psychological Measurement, 20, 37–56.

Collins, M., and N. Duffy (2001), Convolution kernels for natural language, in NIPS, pp. 625–632, MIT Press.

Cover, T. M., and J. A. Thomas (1991), Elements of information theory, Wiley-Interscience, New York, NY, USA.

Craven, M., D. Dipasquo, D. Freitag, A. K. Mccallum, T. M. Mitchell, K. Nigam, and S. Slattery (1998), Learning to extract symbolic knowledge from the world Wide Web, in AAAI, pp. 509–516, AAAI Press, Menlo Park, US, Madison, US.

Cui, P. (2007), A tighter analysis of set cover greedy algorithm for test set, in ESCAPE, pp. 24–35.

Cumby, C. M., and D. Roth (2002), Learning with feature description logics, in International Conference on Inductive Logic Programming, pp. 32–47, Springer-Verlag.

da F. Costa, L., F. A. Rodrigues, G. Travieso, and P. R. V. Boas (2007), Characterization of complex networks: A survey of measurements, Advances In Physics, 56, 167.

Dai, W., and R. Srihari (2005), Minimal document set retrieval, in CIKM, pp. 752–759, ACM, Bremen, Germany.

Daniel, N., D. Radev, and T. Allison (2003), Sub-event based multi-document summarization, HLT-NAACL workshop on Text Summarization, pp. 9–16.

Danushka Bollegala, M. I., Yutaka Matsuo (2007), Measuring semantic similarity between words using web search engines, in WWW, pp. 757–766, ACM Press, New York, NY, USA.

Dasgupta, S., M. L. Littman, and D. Mcallester (2001), PAC generalization bounds for co-training, in NIPS.

Datar, M., and P. Indyk (2004), Locality-sensitive hashing scheme based on p-stable distributions, in SCG, pp. 253–262, ACM Press.

Davidson, I., K. Wagstaff, and S. Basu. (2006), Measuring constraint-set utility for partitional clustering algorithms., in PKDD, pp. 115–126.

Davis, J. V., B. Kulis, P. Jain, S. Sra, and I. S. Dhillon (2007a), Information-theoretic metric learning, in ICML, pp. 209–216, ACM, New York, NY, USA, doi:http://doi.acm.org/10.1145/1273496.1273523.

Davis, J. V., B. Kulis, P. Jain, S. Sra, and I. S. Dhillon (2007b), Information-theoretic metric learning, in ICML, pp. 209–216.

de Solla Price, D. (1970), Citation measures of hard science, soft science, technology, and nonscience, in Communication among Scientists and Engineers, edited by C. E. Nelson and D. K. Pollock, pp. 3–22.

de Solla Price, D. J. (1965), Networks of scientific papers, Science, 149(3683), 510–515.

Deerwester, S., S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman (1990), Indexing by latent semantic analysis, JASIS, 41, 391–407.

Delort, J.-Y., B. Bouchon-Meunier, and M. Rifqi (2003), Enhanced web document summarization using hyperlinks, proceedings of the fourteenth ACM conference on Hypertext and hypermedia, pp. 208p–215.

Demaine, E. D., M. T. Hajiaghayi, U. Feige, and M. R. Salavatipour (2006), Combination can be hard: approximability of the unique coverage problem, in SODA, pp. 162–171, ACM, Miami, Florida.

Demiriz, A., K. Bennett, and M. J. Embrechts (1999), Semi-supervised clustering using genetic algorithms, in In Artificial Neural Networks in Engineering, pp. 809–814, ASME Press.

Dervos, D. A., and T. Kalkanis (2005), cc-IFF: A cascading citations impact factor framework for the automatic ranking of research publications, IEEE International Workshop on Data Acquisition and Advanced Computing Systems Workshop proceedings, pp. 668–673.

Dhillon, I. S., S. Mallela, and D. S. Modha (2003), Information-theoretic co-clustering, in KDD, KDD '03, pp. 89–98, ACM, New York, NY, USA, doi:http://doi.acm.org/10.1145/956750.956764.

Dhillon, I. S., Y. Guan, and B. Kulis (2007), Weighted graph cuts without eigenvectors a multilevel approach, PAMI, 29(11), 1944–1957.

Dice, L. R. (1945), Measures of the amount of ecologic association between species, Ecology, 26(3), 297–302.

Dong, W., M. Charikar, and K. Li (2011), Efficient k-nearest neighbor graph construction for generic similarity measures, in WWW, pp. 577–586.

Doreian, P., and T. J. Fararo (1985), Structural equivalence in a journal network, JASIS, 36, 28–37.

Dorogovtsev, S. N., and J. F. F. Mendes (2002), Evolution of networks, Advances in Physics, 51, 1079.

Druck, G., B. Settles, and A. McCallum (2009), Active learning by labeling features, in EMNLP, pp. 81–90.

Editorial, N. (2005), Not-so-deep impact, Nature, 435(7045), 1003.

Elmacioglu, E., and D. Lee (2005), On six degrees of separation in dblp-db and more, SIGMOD Rec., 34(2), 33–40.

Erdös, P., and A. Rényi (1961), On the evolution of random graphs., Bulletin of the International Statistics Institute, 38, 343–347.

Erkan, G., and D. R. Radev (2004a), LexRank: Graph-based lexical centrality as salience in text summarization, Journal of Artificial Intelligence Research, 22, 457–479.

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

Feige, U. (1998), A threshold of log n for approximating set cover, Journal of the ACM, 45, 634–652.

Ferrer i Cancho, R., and R. V. Solé (2001), The small-world of human language, proceedings of the Royal Society of London B, 268(1482), 2261–2265.

Filatova, E., and V. Hatzivassiloglou (2004), Event-based extractive summarization, ACL Workshop on Summarization, pp. 104–111.

Findler, N., and J. van Leeuwen (1979), A family of similarity measures between two strings, PAMI, 1(1), 116–118.

florina Balcan, M., A. Blum, P. P. Choi, J. Lafferty, B. Pantano, M. R. Rwebangira, and X. Zhu (2005), An application of person identification in webcam images, in ICML Workshop on Learning from Partially Classified Training Data, pp. 1–9.

Fowler, J. H., and D. W. Aksnes (2007), Does self-citation pay?, Scientometrics, 72(3), 427–437.

Frank, A., and A. Asuncion (2010), UCI machine learning repository.

Frank, E., G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning (1999), Domain-specific keyphrase extraction, in IJCAI, pp. 668–673, Morgan Kaufmann Publishers Inc.

Freeman, L. C. (1977), A set of measures of centrality based on betweenness, Sociometry, 40(1), 35–41.

Frey, B. J., and D. Dueck (2007), Clustering by passing messages between data points, Science, 315(5814), 972–976.

Fukuhara, T., T. Murayama, and T. Nishida (2005), Analyzing concerns of people using weblog articles and real world temporal data, in WWW.

Fung, P., G. Ngai, and C.-S. Cheung (2003), Combining optimal clustering and hidden markov models for extractive summarization, ACL workshop on Multilingual summarization and question answering, pp. 21–28.

Gamon, M. (2006), Graph-based text representation for novelty detection, in ACL Workshop on Graph Based Methods for Natural Language Processing, pp. 17–24, Association for Computational Linguistics, New York City.

Ganiz, M. C., N. I. Lytkin, and W. M. Pottenger (2009), Leveraging higher order dependencies between features for text classification, in ECML PKDD '09, Springer-Verlag, Berlin, Heidelberg.

Garey, M. R., and D. S. Johnson (1990), Computers and Intractability; A Guide to the Theory of NP-Completeness, W. H. Freeman.

Garfield, E. (1955), Citation indexes for science: a new dimension in documentation through association of ideas, Science, 122(3159), 108–111.

Garfield, E. (1972), Citation analysis as a tool in journal evaluation, Science, 178(4060), 471–479.

Garfield, E. (1979), Citation Indexing: Its Theory and Application in Science, Wiley, New York.

Getoor, L., and B. Taskar (2007), Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning), The MIT Press.

Giles, C. L., K. D. Bollacker, and S. Lawrence (1998), CiteSeer: An automatic citation indexing system, in DL, pp. 89–98, ACM, Pittsburgh, Pennsylvania, United States.

Gionis, A., P. Indyk, and R. Motwani (1999), Similarity search in high dimensions via hashing, in VLDB, pp. 518–529, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Glänzel, W. (2004), Towards a model for diachonous and synchronous citation analyses, Scientometrics, 60(3), 511–522.

Goldberg, A. B., X. Zhu, and S. Wright (2007), Dissimilarity in graph-based semi-supervised classification, Journal of Machine Learning Research, 2, 155–162.

Golub, G., and C. Loan (1996), Matrix computations, Johns Hopkins studies in the mathematical sciences, Johns Hopkins University Press.

Golub, G., and C. Reinsch (1970), Singular value decomposition and least squares solutions, Numerische Mathematik, 14, 403–420.

Goodrum, A. A., K. W. McCaina, S. Lawrence, and C. L. Giles (2001), Scholarly publishing in the Internet age: A citation analysis of computer science literature, IPM, 37(6), 661–675.

Griffith, B. C., and P. N. Servi (1980), A method for partitioning the journal literature, JASIS, 31, 36–40.

Grira, N., M. Crucianu, and N. Boujemaa (), Unsupervised and semi-supervised clustering: a brief survey, in The Review of Machine Learning Techniques for Processing Multimedia Content, Report of the MUSCLE European Network of Excellence (FP6), year = 2005.

Gruhl, D., R. Guha, D. Liben-Nowell, and A. Tomkins (2004), Information diffusion through blogspace, pp. 491–501, ACM, New York, NY, USA.

Hall, D., D. Jurafsky, and C. Manning (2008), Studying the history of ideas using topic models, in EMNLP.

Harel, D., and Y. Koren (2001), On clustering using random walks, in Foundations of Software Technology and Theoretical Computer Science 2245, pp. 18–41, Springer-Verlag.

Harzing, A.-W. (2008), Publish or perish version 2.5 (software), http://www.harzing.com/pop.htm.

Hassan, A., and D. R. Radev (2010), A mixture of networks approach to combining link based and content based networks, in Preparation.

He, X., C. H. Q. Ding, H. Zha, and H. D. Simon (2001), Automatic topic identification using webpage clustering, ICDM, pp. 195–202.

Heyer, L. J., S. Kruglyak, and S. Yooseph (1999), Exploring expression data: Identification and analysis of coexpressed genes., Genome Research, 9, 1106–1115.

Hirsch, J. E. (2005), An index to quantify an individual's scientific research output, PNAS, 102, 16,569.

Hirsch, J. E. (2007), Does the h-index have predictive power?, PNAS, 104, 191–193.

Hofmann, T. (1999a), Probabilistic latent semantic analysis, in UAI, pp. 289–296.

Hofmann, T. (1999b), Probabilistic latent semantic indexing, in SIGIR, pp. 50–57, ACM, New York, NY, USA.

Hoopdog (2007), Follow-up: Blame game. http://hoopdogg.livejournal.com/39060.html.

Hu, M., A. Sun, and E.-P. Lim (2007), Comments-oriented blog summarization by sentence extraction, pp. 901–904.

Hull, J. J. (1994), A database for handwritten text recognition research, PAMI, 16(5), 550–554.

Hulth, A. (2003), Improved automatic keyword extraction given more linguistic knowledge, EMNLP, pp. 216–223.

Ikeda, D., T. Fujiki, and M. Okumura (2006), Automatically linking news articles to blog entries.

Jaccard, P. (1901), Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines, Bulletin de la Société Vaudoise des Sciences Naturelles, 37, 241–272.

Jain, A. K., and R. C. Dubes (1988), Algorithms for Clustering Data., Prentice-Hall.

Jeh, G., and J. Widom (2002), Simrank: a measure of structural-context similarity, in KDD, pp. 538–543, ACM Press, New York, NY, USA.

Jeong, H., S. Mason, A. Barabási, and Z. Oltvai (2001), Lethality and centrality in protein networks, Nature, 411(6833), 41–42.

Jiang, J. J., and D. W. Conrath (1997), Semantic similarity based on corpus statistics and lexical taxonomy, in ROCLING, pp. 9008+.

Joachims, T. (1998), Making large-scale support vector machine learning practical, MIT Press, Cambridge, MA.

Jones, K. S. (1972), A statistical interpretation of term specificity and its application in retrieval, Journal of Documentation, 28, 11–21.

Joseph, M. T., and D. R. Radev (2007), Citation analysis, centrality, and the ACL anthology, Tech. Rep. CSE-TR-535-07, Department of Computer Science and Engineering, University of Michigan.

Karov, Y., and S. Edelman (1998), Similarity-based word sense disambiguation, Computational Linguistics, 24, 41–59.

Karypis, G., and V. Kumar (1998), A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM Journal on Scientific Computing, 20, 359–392.

Kashyap, R., and B. Oommen (1983), The noisy substring matching problem, SE, 9, 365–370.

Kestler, H. A., A. Muller, T. M. Gress, and M. Buchholz (2005), Generalized venn diagrams: a new method of visualizing complex genetic set relations, Bioinformatics, 21, 1592–1595, doi:10.1093/bioinformatics/bti169.

Khuller, S., A. Moss, and J. S. Naor (1999), The budgeted maximum coverage problem, Information Processing Letters, 70(1), 39–45.

Kim, W., and W. J. Wilbur (2001), Corpus-based statistical screening for content-bearing terms, J. Am. Soc. Inf. Sci. Technol., 52, 247–259.

Kleinberg, J. M. (1999), Authoritative sources in a hyperlinked environment, Journal of the ACM, 46(5), 604–632.

Kontostathis, A., and W. M. Pottenger (2006), A framework for understanding latent semantic indexing (lsi) performance, IPM, 42(1), 56–73.

Koren, Y. (2008), Factorization meets the neighborhood: a multifaceted collaborative filtering model, in KDD, KDD '08, pp. 426–434, ACM, New York, NY, USA.

Koschützki, D., K. A. Lehmann, D. Tenfelde-Podehl, and O. Zlotowski (2005), Advanced centrality concepts, pp. 83–111.

Kraaij, W., and M. Spitters (2001), TNO at TDT 2001: Language model-based topic detection, Topic Detection and Tracking Workshop Report.

Krovetz, R. (1993), Viewing morphology as an inference process, in SIGIR, pp. 191–202.

Ku, L.-W., L.-Y. Lee, T.-H. Wu, and H.-H. Chen (2005), Major topic detection and its application to opinion summarization, SIGIR, pp. 627–628.

Kulis, B., S. Basu, I. Dhillon, and R. Mooney (2009), Semi-supervised graph clustering: a kernel approach, Machine Learning, 74(1), 1–22.

Lanckriet, G., N. Cristianini, P. Bartlett, and L. E. Ghaoui (2002), Learning the kernel matrix with semi-definite programming, Journal of Machine Learning Research, 5, 2004.

Lawrence, P. A. (2003), The politics of publication, Nature, 422, 259–261.

Lawrence, S. (2001), Online or invisible?, Nature, 411(6837), 521.

Lawrence, S., C. L. Giles, and K. Bollacker (1999), Digital libraries and autonomous citation indexing, 32(6), 67–71.

Leacock, C., and M. Chodorow (1998), Combining local context and wordnet similarity for word sense identification, An Electronic Lexical Database, pp. 265–283.

LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1990), Handwritten digit recognition with a back-propagation network, in Advances in Neural Information Processing Systems, pp. 396–404, Morgan Kaufmann.

Lee, I., S. V. Date, A. T. Adai, and E. M. Marcotte (2004), A probabilistic functional network of yeast genes, Science, 306(5701), 1555–1558.

Lee, L. (1999), Measures of distributional similarity, ACL, pp. 25–32.

Lehmann, S., A. D. Jackson, and B. E. Lautrup (2006), Measures for measures, Nature, 444, 1003–1004.

Leicht, E., G. Clarkson, K. Shedden, and M. E. J. Newman (2007), Large-scale structure of time evolving citation networks, The European Physical Journal B, 59, 75.

Leskovec, J., A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance (2007), Cost-effective outbreak detection in networks, in KDD, pp. 420–429, ACM, New York, NY, USA.

Leslie, C., and R. Kuang (2004), Fast string kernels using inexact matching for protein sequences, Journal of Machine Learning Research, 5, 1435–1455.

Leslie, C. S., E. Eskin, and W. S. Noble (2002), The spectrum kernel: A string kernel for svm protein classification, in Pacific Symposium on Biocomputing, pp. 566–575.

Levenshtein, V. (1966a), Binary codes capable of correcting deletions, insertions, and reversals, Soviet Physics Doklady, 10(8), 707–710.

Levenshtein, V. I. (1966b), Binary codes capable of correcting deletions, insertions, and reversals, Soviet Physics Doklady, 10(8), 707–710.

Leydesdorff, L. (1998), Theories of citation?, Scientometrics, 43(1), 5–25.

Leydesdorff, L. (2002), Indicators of structural change in the dynamics of science: Entropy statistics of the sci journal citation reports, Scientometrics, 53(1), 131–159.

Leydesdorff, L. (2003), Can networks of journal-journal citations be used as indicators of change in the social sciences?, Journal of Documentation, 59(1), 84–104.

Leydesdorff, L. (2004a), Clusters and maps of science journals based on bi-connected graphs in the journal citation reports, Journal of Documentation, 60(4), 371–427.

Leydesdorff, L. (2004b), Top-down decomposition of the journal citation report of the social science citation index: Graph- and factor-analytical approaches, Scientometrics, 60(2), 159–180.

Leydesdorff, L. (2005), Similarity measures, author cocitation analysis, and information theory, JASIS, 56(7), 769–772.

Leydesdorff, L. (2006), Can scientific journals be classified in terms of aggregated journal-journal citation relations using the journal citation reports?, JASIS, 57(5), 601–613.

Leydesdorff, L., and S. J. Bensman (2006), Classification and powerlaws: The logarithmic transformation, JASIS, 57(11), 1470–1486.

Leydesdorff, L., and S. E. Cozzens (1993), The delineation of specialties in terms of journals using the dynamic journal set of the science citation index, Scientometrics, 26, 133–154.

Leydesdorff, L., and B. Jin (2005), Mapping the chinese science citation database in terms of aggregated journal-journal citation relations, JASIS, 56(14), 1469–1479.

Leydesdorff, L., and P. van der Schaar (1987), The use of scientometric indicators for evaluating national research programmes, Science and Technology Studies, 5, 22–31.

Leydesdorff, L., and L. Vaughan (2006), Co-occurrence matrices and their applications in information science: Extending ACA to the web environment, JASIS, 57(12), 1616–1628.

Leydesdorff, L., S. E. Cozzens, and P. van den Besselaar (1994), Tracking areas of strategic importance using scientometric journal mappings, Research Policy, 23, 217–229.

Li, W., M. Wu, Q. Lu, W. Xu, and C. Yuan (2006), Extractive summarization using inter- and intra-event relevance, COLING-ACL, pp. 369–376, doi:10.3115/1220175.1220222.

Liddy, E. (2001), Advances in automatic text summarization, Inf. Retr, 4, 82–83.

Liu, T., A. W. Moore, A. Gray, and K. Yang (2004), An investigation of practical approximate nearest neighbor algorithms, pp. 825–832, MIT Press.

Liu, X., and W. B. Croft (2004), Cluster-based retrieval using language models, SIGIR, pp. 186–193.

Lodhi, H., C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins (2002), Text classification using string kernels, Journal of Machine Learning Research, 2, 419–444.

Luo, Z. Q., and P. Tseng (1992), On the convergence of the coordinate descent method for convex differentiable minimization, Journal of Optimization Theory and Applications, 72(1), 7–35.

MacKay, D. J. C., and L. Peto (1994), A hierarchical dirichlet language model, Natural Language Engineering, 1, 1–19.

MacQueen, J. (1967), Some methods for classification and analysis of multivariate observations, in proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297.

Mahalanobis, P. C. (1936), On the generalised distance in statistics, in Proceedings of National Institute of Science, India, vol. 2, pp. 49–55.

Mallapragada, P. K., R. Jin, and A. K. Jain (2008), Active query selection for semi-supervised clustering., in ICPR, pp. 1–4, IEEE.

Mann, G. S., D. Mimno, and A. McCallum (2006), Bibliometric impact measures leveraging topic analysis, in JCDL, pp. 65–74, Chapel Hill, NC, USA.

Markov, I., L. Scheffer, and S. Hufnagel (2007), DUplicate text DEtection, or DUDE, http://sigda.eecs.umich.edu/DUDE/.

McCallum, A. K. (2002), Mallet: A machine learning for language toolkit, http://mallet.cs.umass.edu.

Mei, Q., and C. Zhai (2005), Discovering evolutionary theme patterns from text: an exploration of temporal text mining, KDD, pp. 198–207.

Mei, Q., X. Shen, and C. Zhai (2007), Automatic labeling of multinomial topic models, KDD, pp. 490–499.

Mei, Q., D. Cai, D. Zhang, and C. Zhai (2008), Topic modeling with network regularization, in WWW, pp. 101–110, ACM, New York, NY, USA.

Metzler, D., S. T. Dumais, and C. Meek (2007), Similarity measures for short segments of text., in ECIR, LNCS, vol. 4425, edited by G. Amati, C. Carpineto, and G. Romano, pp. 16–27, Springer.

Mihalcea, R., and C. Corley (2006), Corpus-based and knowledge-based measures of text semantic similarity, in AAAI, pp. 775–780.

Milgram, S. (1967), The small world problem, Psychology Today, pp. 60–67.

Miller, G. A. (1995), Wordnet: A lexical database for english, Communications of the ACM, 38, 39–41.

Miller, G. A., and W. G. Charles (1991), Contextual correlates of semantic similarity, Language and Cognitive Processes, 6(1), 1–28.

Mishne, G. (2007), Applied text analytics for blogs, Ph.D. thesis, PhD thesis, University of Amsterdam.

Moschitti, A. (2004), A study on convolution kernels for shallow semantic parsing, in ACL, p. 335, Association for Computational Linguistics, Morristown, NJ, USA, doi:http://dx.doi.org/10.3115/1218955.1218998.

Moschitti, A. (2006), Making tree kernels practical for natural language learning, in EACL.

Muthukrishnan, P., J. Gerrish, and D. R. Radev (2008), Detecting multiple facets of an event using graph-based unsupervised methods, in COLING, pp. 609–616.

Muthukrishnan, P., D. R. Radev, and Q. Mei (2010), Edge weight regularization over multiple graphs for similarity learning, in ICDM, pp. 374–383, IEEE Computer Society.

Nakatsu, N., Y. Kambayashi, and S. Yajima (1982), A longest common subsequence algorithm suitable for similar text strings, Acta Inf., 18, 171–179.

Narin, F., M. Carpenter, and N. C. Berlt (1972), Interrelationships of scientific journals, JASIS, 23, 323–331.

Nascimento, M. A., J. Sander, and J. Pound (2003), Analysis of SIGMOD's coauthorship graph, ACM SIGMOD Record, 32(3).

Nederhof, A. J., R. A. Zwaan, R. E. Bruin, and P. J. Dekker (1989), Assessing the usefulness of bibliometric indicators for the humanities and the social sciences: A comparative study, Scientometrics, 15, 423–436.

Nemhauser, G. L., L. Wolsey, and M. Fisher (1978), An analysis of approximations for maximizing submodular set functions, Mathematical Programming, 14.

Newman, M. E. J. (2001), Who is the best connected scientist? a study of scientific coauthorship networks. part ii. shortest paths, weighted networks, and centrality, PRE, 64(1), 016,131.

Newman, M. E. J. (2003a), A measure of betweenness centrality based on random walks, Tech. Rep. cond-mat/0309045, Arxiv.org.

Newman, M. E. J. (2003b), The structure and function of complex networks, SIAM Review, 45, 167–265.

Newman, M. E. J. (2003c), The structure and function of complex networks, SIAM Review, 45(2), 167–256.

Newman, M. E. J. (2004), Coauthorship networks and patterns of scientific collaboration, PNAS, 101, 5200–5205.

Newman, M. E. J. (2005), Power laws, Pareto distributions and Zipf's law, Contemporary Physics, 46(5), 323–351.

Newman, M. E. J., D. J. Watts, and S. H. Strogatz (2002), Random graph models of social networks, PNAS, 99, 2566–2572, suppl.1.

Nigam, K., and R. Ghani (2000), Analyzing the effectiveness and applicability of co-training, in CIKM, pp. 86–93, ACM, New York, NY, USA, doi:http://doi.acm.org/10.1145/354756.354805.

Ogata, H., S. Goto, K. Sato, W. Fujibuchi, H. Bono, and M. Kanehisa (1999), Kegg: Kyoto encyclopedia of genes and genomes, Nucleic Acids Research, 27(1), 29–34.

Oka, M., H. Abe, and K. Kato (2006), Extracting topics from weblogs through frequency segments, WWW.

Osman, D. J., and J. L. Yearwood (2007), Opinion search in web logs, in ECAD, pp. 133–139, Australian Computer Society, Inc, Ballarat, Victoria, Australia.

Page, L., S. Brin, R. Motwani, and T. Winograd (1998), The pagerank citation ranking: Bringing order to the web, Tech. rep., Stanford Digital Library Technologies Project.

Papineni, K., S. Roukos, T. Ward, and W. jing Zhu (2002), Bleu: A method for automatic evaluation of machine translation, pp. 311–318.

Parrott, W. G. (2001), Emotions in Social Psychology: Essential Readings, Psychology Press.

Pauly, D., and K. Stergiou (2005), Equivalence of results from two citation analyses: Thomson ISI's Citation Index and Google's Scholar service, Ethics in Science and Environmental Politics, pp. 33–35.

Pinski, G., and F. Narin (), Citation influence for journal aggregates of scientific publications: Theory, with application to the literature of physics, IPM.

Ponte, J. M., and W. B. Croft (1998), A language modeling approach to information retrieval, pp. 275–281.

Porter, M. F. (1997), An algorithm for suffix stripping, Program, pp. 313–316.

Radev, D. R., and K. R. McKeown (1998), Generating natural language summaries from multiple on-line sources, Computational Linguistics, 24, 470–500.

Radev, D. R., and D. Tam (2003), Single-document and multi-document summary evaluation via relative utility, in CIKM.

Radev, D. R., H. Jing, and M. Budzikowska (2000), Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies, NAACL-ANLP 2000 Workshop on Automatic summarization, pp. 21–30.

Radev, D. R., S. Blair-Goldensohn, Z. Zhang, and R. S. Raghavan (2001a), Newsinessence: a system for domain-independent, real-time news clustering and multi-document summarization, pp. 1–4, Association for Computational Linguistics, San Diego.

Radev, D. R., W. Fan, and Z. Zhang (2001b), Webinessence: a personalized web-based multi-document summarization and recommendation system, in In NAACL Workshop on Automatic Summarization, pp. 79–88.

Radev, D. R., M. Hodges, A. Fader, M. Joseph, J. Gerrish, M. Schaller, J. dePeri, and B. Gibson (2007), Clairlib documentation v1.03, Tech. Rep. CSE-TR-536-07, Department of Computer Science and Engineering, University of Michigan.

Radev, D. R., M. T. Joseph, B. Gibson, and P. Muthukrishnan (2009a), A Bibliometric and Network Analysis of the field of Computational Linguistics, JASIS.

Radev, D. R., P. Muthukrishnan, and V. Qazvinian. (2009b), The ACL Anthology Network corpus., in the ACL Workshop on Natural Language Processing and Information Retrieval for Digital Libraries.

Raghavan, H., O. Madani, R. Jones, and P. Kaelbling (2006), Active learning with feedback on both features and instances, Journal of Machine Learning Research, 7.

Rahm, E., and A. Thor (2005), Citation analysis of database publications, in SIGMOD, vol. 34.

Redner, S. (1998), How popular is your paper? An empirical study of the citation distribution, European Physical Journal B, 4(2), 131–134.

Resnik, P. (1995), Using information content to evaluate semantic similarity in a taxonomy, in IJCAI, pp. 448–453.

Resnik, P., and N. A. Smith (2003), The web as a parallel corpus, Computational Linguistics, 29, 349–380.

Rocklin, M., and A. Pinar (2011), On clustering on graphs with multiple edge types, ArXiv e-prints.

Rosen, N. D. (2007), Gun control and mental health. http://ndrosen.livejournal.com/128715.html.

Rousseau, R., and A. Zuccala (2004), A classification of author co-citations: Definitions and search strategies, JASIS, 55(6), 513–529.

Rubenstein, H., and J. B. Goodenough (1965), Contextual correlates of synonymy, Communications of ACM, 8, 627–633.

Sahami, M., and T. D. Heilman (2006), A web-based kernel function for measuring the similarity of short text snippets, in WWW, pp. 377–386, ACM Press, New York, NY, USA.

Salton, G., and M. J. McGill (1986), Introduction to Modern Information Retrieval, McGraw-Hill, Inc. New York, NY, USA.

Salton, G., A. Wong, and C. S. Yang (1975), A vector space model for automatic indexing, Communications of the ACM, 18(11), 613–620.

Saunders, C., H. Tschach, and J. Shawe-Taylor (2002), Syllables and other string kernel extensions, in ICML.

Schultz, M., and T. Joachims (2003), Learning a distance metric from relative comparisons, in NIPS, MIT Press.

Schütze, H. (1998), Automatic word sense discrimination, Computational Linguistics, 24(1), 97–123.

Sebastiani, F. (2002), Machine learning in automated text categorization, ACM Computing Surveys, 34, 1–47.

Sen, P., G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad (2008), Collective classification in network data, AI Magazine, 29(3), 93–106.

Settles, B. (2009), Active learning literature survey, Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Shi, J., and J. Malik (2000), Normalized cuts and image segmentation, PAMI, 22, 888–905.

Shi, X., B. Tseng, and L. Adamic (2009), Information diffusion in computer science citation networks, in ICWSM.

Sichel, H. S. (1985), A bibliometric distribution which really works, JASIS, 36(5), 314–321.

Sidiropoulos, A., D. Katsaros, and Y. Manolopoulos (2006), Generalized h-index for disclosing latent facts in citation networks, http://arxiv.org/abs/cs.DL/0607066 visited on June 18, 2008.

Silagadze, Z. (1997), Citations and the Zipf-Mandelbrot's law, Complex Systems, 11, 487–499.

Small, H. (1973), Co-citation in the scientific literature: A new measure of the relationship between two documents, JASIS, 24, 265–269.

Small, H. (1999), Visualizing science by citation mapping, JASIS, 50, 799–813.

Stone, P. J. (1966), The General Inquirer: A Computer Approach to Content Analysis, MIT Press.

Strehl, A., and J. Ghosh (2002), Cluster ensembles: a knowledge reuse framework for combining partitionings, in AAAI, pp. 93–98, American Association for Artificial Intelligence, Menlo Park, CA, USA.

Strehl, A., J. Ghosh, and R. Mooney (2000), Impact of similarity measures on web-page clustering, in AAAI Workshop of Artificial Intelligence for Web Search, pp. 58–64, AAAI.

Strube, M., and S. P. Ponzetto (2006), Wikirelate! computing semantic relatedness using wikipedia, in AAAI, AAAI Press.

Teo, C. H., and S. V. N. Vishwanathan (2006), Fast and space efficient string kernels using suffix arrays, in ICML, pp. 929–936, ACM, New York, NY, USA.

Tomokiyo, T., and M. Hurst (2003), A language model approach to keyphrase extraction, ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment, pp. 33–40.

Turney, P. D. (2000), Learning algorithms for keyphrase extraction, Information Retrieval, 2, 303–336.

Turney, P. D. (2001), Mining the web for synonyms: PMI-IR versus LSA on TOEFL, in ECML, pp. 491–502.

Tversky, A. (1977), Features of similarity, in Psychological Review, vol. 84, pp. 327–352.

van Eck, N. J., and L. Waltman (2009), How to normalize cooccurrence data. an analysis of some well-known similarity measures, JASIS, 60, 1635–1651, doi:10.1002/asi.v60:8.

Vishwanathan, S. V. N., and A. Smola (2003), Fast kernels for string and tree matching, in NIPS, vol. 15, edited by S. Becker, S. Thrun, and K. Obermayer, MIT Press.

Wagstaff, K. (2006), Value, cost, and sharing: Open issues in constrained clustering, in KDD, pp. 1–10.

Watts, D. J. (2003), Six Degrees: The Science of a Connected Age, W. W. Norton & Company.

Watts, D. J., and S. H. Strogatz (1998), Collective dynamics of small-world networks, Nature, 393(6684), 440–442.

Wellner, B., A. McCallum, F. Peng, and M. Hay (2004), An integrated, conditional model of information extraction and coreference with application to citation matching, in UAI.

Wilson, T., J. Wiebe, and P. Hoffmann (2005), Recognizing contextual polarity in phrase-level sentiment analysis, pp. 347–354, Association for Computational Linguistics, Vancouver, British Columbia, Canada.

Witten, I. H., G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning (1999), Kea: practical automatic keyphrase extraction, in CDL, pp. 254–255, Berkeley, California, United States.

Xu, Yuan, Li, Wu, and Wong (2006), Building document graphs for multiple news articles summarization: An event-based approach, in ICCPOL.

Xu, J., and W. B. Croft (1998), Corpus-based stemming using cooccurrence of word variants, ACM Transactions on Information Systems, 16(1), 61–81.

Yahoo! (2011), Yahoo! kdd cup.

York, D. (1966), Least-square fitting of a straight line, Canadian Journal of Phys, 44, 1079–1086.

Zhai, C., and J. Lafferty (2004), A study of smoothing methods for language models applied to information retrieval, ACM Trans. Inf. Syst., 22, 179–214.

Zhai, C. X., W. W. Cohen, and J. Lafferty (2003), Beyond independent relevance: methods and evaluation metrics for subtopic retrieval, in SIGIR, pp. 10–17, ACM, Toronto, Canada.

Zhang, Y., J. P. Callan, and T. P. Minka (2002), Novelty and redundancy detection in adaptive filtering, in SIGIR, pp. 81–88.

Zhou, D., and C. J. C. Burges (2007), Spectral clustering and transductive learning with multiple views, in ICML, pp. 1159–1166, ACM, New York, NY, USA.

Zhou, D., S. Zhu, K. Yu, X. Song, B. L. Tseng, H. Zha, and C. L. Giles (2008), Learning multiple graphs for document recommendations, in WWW, pp. 141–150, ACM, New York, NY, USA.

Zhou, L., and E. Hovy (2006), On the summarization of dynamically introduced information: Online discussions and blogs, in AAAI Symposium on Computational Approaches to Analysing Weblogs, pp. 237–242.

Zhu, X., J. Lafferty, and Z. Ghahramani (2003a), Combining active learning and semi-supervised learning using gaussian fields and harmonic functions, in ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining, pp. 58–65.

Zhu, X. Z., Z. Ghahramani, and J. Lafferty (2003b), Semi-supervised learning using gaussian fields and harmonic functions, in ICML, pp. 912–919.