Using Dependency Parses to Augment Feature Construction for Text Mining

Sheng Guo

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science

Naren Ramakrishnan, Chair
Edward A. Fox
Richard Helm
T. M. Murali
Mohammed J. Zaki

May 2, 2012
Blacksburg, Virginia

Keywords: Dependency parsing, text mining, linguistic cues.

Using Dependency Parses to Augment Feature Construction for Text Mining

Sheng Guo

(ABSTRACT)

With the prevalence of large data stored in the cloud, including unstructured information in the form of text, there is now an increased emphasis on text mining. A broad range of techniques are now used for text mining, including algorithms adapted from machine learning, NLP, computational linguistics, and data mining. Applications are also multi-fold, including classification, clustering, segmentation, relationship discovery, and practically any task that discovers latent information from written natural language.

Classical mining algorithms have traditionally focused on shallow representations such as bag-of-words and similar feature-based models. With the advent of modern high performance computing, deep sentence level linguistic analysis of large scale text corpora has become practical. In this dissertation, we evaluate the utility of dependency parses as textual features for different text mining applications. Dependency parsing is one form of syntactic parsing, based on the dependency grammar implicit in sentences. While dependency parsing has traditionally been used for text understanding, we investigate here its application to supply features for text mining applications.

We specifically focus on three methods to construct textual features from dependency parses. First, we consider a dependency parse as a general feature akin to a traditional bag-of-words model. Second, we consider the dependency parse as the basis to build a feature graph representation. Finally, we use dependency parses in a supervised collocation mining method for feature selection. To investigate these three methods, several applications are studied, including: (i) movie spoiler detection, (ii) text segmentation, (iii) query expansion, and (iv) recommender systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Text mining applications [2, 60, 111] focus on discovering hidden or latent information from written natural language text, by integrating techniques from the broader data mining arena and from natural language processing. For the purposes of this thesis, a general architecture of a text mining system can be conceptualized as shown in Fig. 1.1. Information extraction is a key step that captures useful information from the originally semi-structured text sources, and supplies many kinds of features for later processing by data mining algorithms. These features typically are drawn from different levels of analysis, e.g., the word/token level, the syntactic level or even the semantic level. Token level analysis focuses on single or multiple words, and studies their properties. Techniques such as tokenization, stemming, and part-of-speech tagging apply here. Syntactic analysis extracts syntactic relationships between tokens. Semantic analysis pertains to distilling more complex and richer forms of information, including ontological and descriptive logics.

A key tradeoff in text mining lies between the sophistication of feature extraction and the complexity of the underlying extraction algorithms. In this dissertation, we argue that dependency parses [76] are a key 'sweet spot' in this tradeoff, i.e., they provide sufficiently discriminative features for text mining but are not too complex to compute/extract. Our work is the first to focus on dependency parses as the source for a key class of features.

With the advent of modern high performance computers, deep sentence-level linguistic analysis for large scale text corpora has become practical. Dependency-based syntactic parsing has gained increasing interest in recent years. This technique reads a sentence as input, and outputs a dependency graph structure. We will discuss this technique in detail in later chapters but see Figure 1.2 for a preview example. In contrast to a vector-space 'bag-of-words' (BOW) model for the whole document, or even an *n*-gram model, dependency parses are analyzed sentence by sentence, and have the following advantages:

1. Dependency parses preserve grammatical relations between terms, whereas word-level features such as BOW and *n*-gram analysis lose connections between terms.

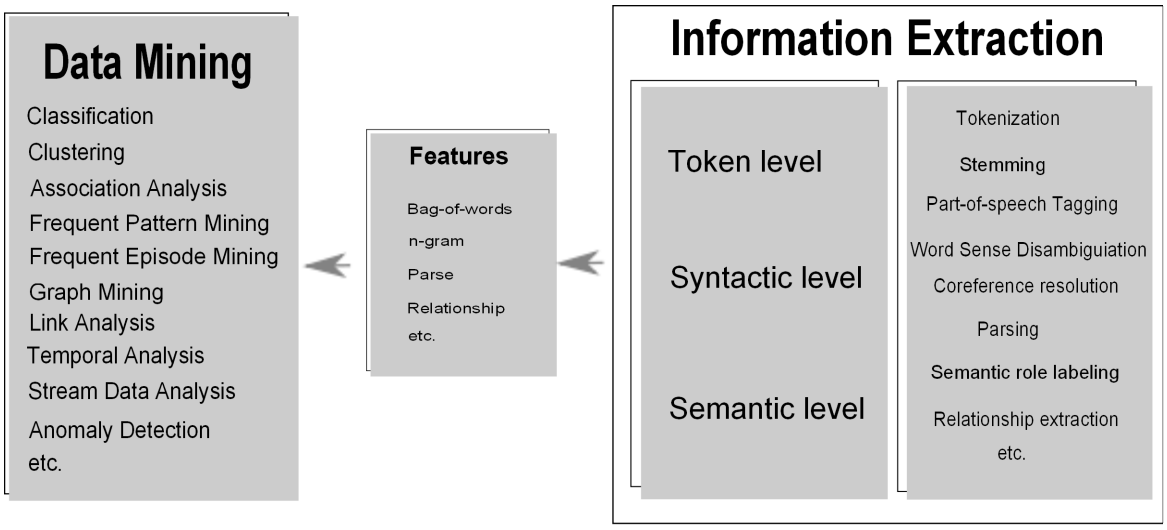


Figure 1.1: Architecture of a text mining system.

2. Dependency trees are extracted in the sentence level and preserved more grammatical information, so large scale datasets are typically not necessary. However, statistical methods, such as *n*-grams, only extract frequency information from each sentence, hence it usually requires a significant amount of data.

3. Dependency parsing often involves word-level linguistic analysis and thus part-of-speech information is also inferrable after parsing.

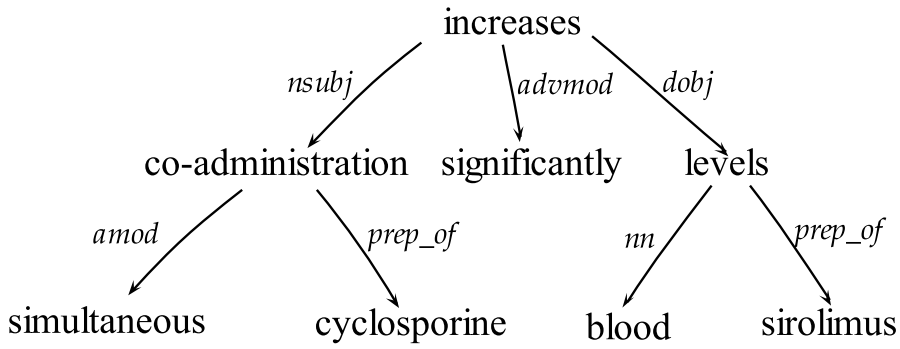


Figure 1.2: Dependency parse for the sentence “Simultaneous co-administration of cyclosporine significantly increases blood levels of sirolimus”.

This document evaluates the use of dependency parsing to supply deeper syntactic features for text mining applications. We specifically focus on five applications (see Fig. 1.3): (i) query expansion,

(ii) topic modeling, (iii) similarity modeling, (iv) text segmentation, and (v) collaborative filtering. The questions we seek to study are whether the use of these more complex features gives significant benefits for machine learning.

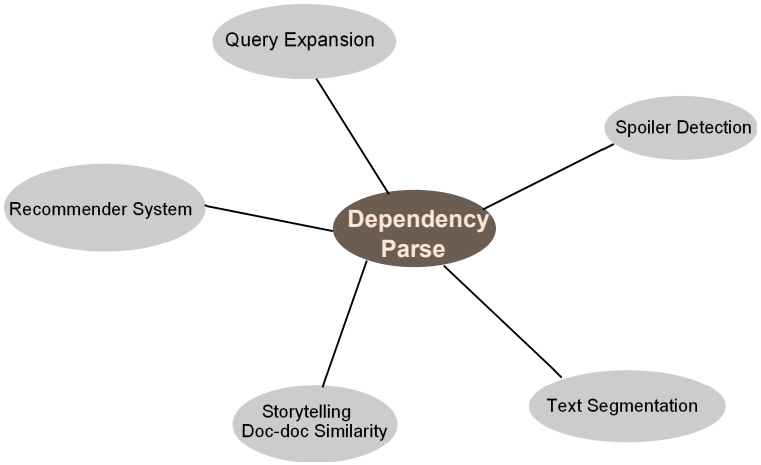


Figure 1.3: Applications of dependency parsing explored in this document.

1.1 From Dependency Parsing to Text Mining

In this document, we focus on applications of dependency parses. To use dependency parsing in text mining, we connect them by extracting helpful textual features and using them in different ways. We study three broad approaches toward these goals.

1.1.1 Dependency Parse as a General Feature

First, well-chosen dependency parses can be used as textual features in a direct way. We study the spoiler detection problem to demonstrate how dependency parses can enhance the underlying text representation for this problem. As a supplement to the conventional bag-of-words model and n-gram representation, dependency parses are more powerful to capture text semantics. Well-chosen dependency parses, by specifying the relation type or the part-of-speech tag, retain more information about sentences than individual terms bundled into a vector-space model. Similar to keywords or phrases in text, such dependency parses have strong specificity to capture semantics. Furthermore, dependency parses are results of syntactic analysis, do not require complicated text processing, are more general, and are widely adopted.

As a general feature, well-chosen dependency parses can be integrated with other features to perform topic analysis. In the spoiler detection problem, we show how a mixture of both dependency

parse and BOW features models the topic information in a better way when doing LDA (Latent Dirichlet Allocation) analysis. A naive method to detect spoilers could be using the plot keywords, or directly searching for the well-known warning messages in the comments, such as “spoiler ahead”. Nonetheless, due to the diversity of linguistic structure and massive noise, the accuracy is usually not adequate. We apply LDA with linguistic dependency information to solve the spoiler detection problem based on predictive perplexity values.

1.1.2 Dependency Parse as a Graph

Secondly, the dependency parse can be viewed as a summary of relationships between terms, and thus can be used to construct a term-term graph. Rather than treating dependency parses as a general feature above, we establish the term graph for text snippets by connecting their dependency parses. While dependency parses as general features are typically associated with vector-space models, a graph representation of text allows graph techniques to be used in text analysis. An example is using graph similarity to model textual similarity, and then linearly segmenting text using breaks in similarity values.

1.1.3 Supervised Collocation Mining

Finally, in addition to the two techniques above, we also introduce a supervised collocation mining method based on textual features. Collocation is a linguistic concept introduced by J. R. Firth [40]. It is widely used in corpus linguistics to identify frequently recurring word combinations. Collocation extraction has been widely used in the areas of natural language generation, computational lexicography, word sense disambiguation, and language teaching. In our study, we define a collocation as “two or more syntactically related words” which have a tendency to co-occur with each other through a dependency relation. Although the original collocations from linguistics are extracted from large natural language corpora, we borrowed this idea and seek a relaxed concept of “*multi-word*” expression, rather than strict idiomatic structures.

In general, when textual features are used in text mining, those features with discriminative ability are more important than others, especially for text classification applications. Supervised collocation mining aims to extract those collocations with such discriminative properties. With large scale training datasets, we can identify discriminative collocations by analyzing their distribution in both positive data and negative data. In this document, we study methods to extract discriminative collocations and their applications.

The first application scenario involves query expansion for domain-specific search. When searching for drug interactions in a large textual corpus, appending the query terms with discriminative expansions can help to find the drug interactions between the specified drugs in these query terms. A very straightforward way is to generate the expanding query terms based on term-term relationships. However, not all the relevant terms for expansion co-occur frequently with the original term.

Meanwhile, terms co-occurring with the original term may not be ideal for query expansion, as pointed out by [121]. For the drug interaction modeling problem, such traditional query expansion methods do not perform well. Given a drug name as the original query, simply using the term-term relationship or co-occurrence information cannot generate high-quality expanding query terms for drug interaction identification. Our task falls into the same category of domain/scenario-specific query expansion as in [43, 94, 118].

In contrast to general web search, domain-specific query expansion requires a domain-relevant expanding query term instead of one related to the original term. [94] considers the typical scenario of searching for treatments for a given disease term from a medical text corpus; traditional query expansion is not suitable here since expanding terms based on the correlation with the initial disease term may not help in identifying terms (e.g., the traditional query expansion may generate *lung excision* for the original query of *lung cancer*, but this will aid in searching for treatments). By conducting supervised collocation mining, we use both positive and negative training sets to bridge the gap between collocation techniques and the domain-specific requirement, thus find better candidate expansion terms for drug interaction search.

We also applied supervised collocation mining in the context of a movie recommender system. As we explained above, discriminative dependency parses can be extracted from the training data which consists of both positive and negative movie comments. These domain-specific sentiment features could be applied to enhance a general recommender system.

The three methods to use dependency parses mentioned above and their applications can be depicted as shown in Fig. 1.4. When dependency parse is used as a general feature, it is suited for analysis of textual data in any size. A graph representation is more suited for short text since it retains more information than statistical method. Simply relying on statistical analysis performs not very well when text is not that long. Finally, supervised collocation mining is used when our goal is discriminative keyword extraction. In summary, considering the differences between all these applications, dependency parses are suitable for different contexts.

In the following chapters, we will look into the details of these applications, and their design and evaluation.

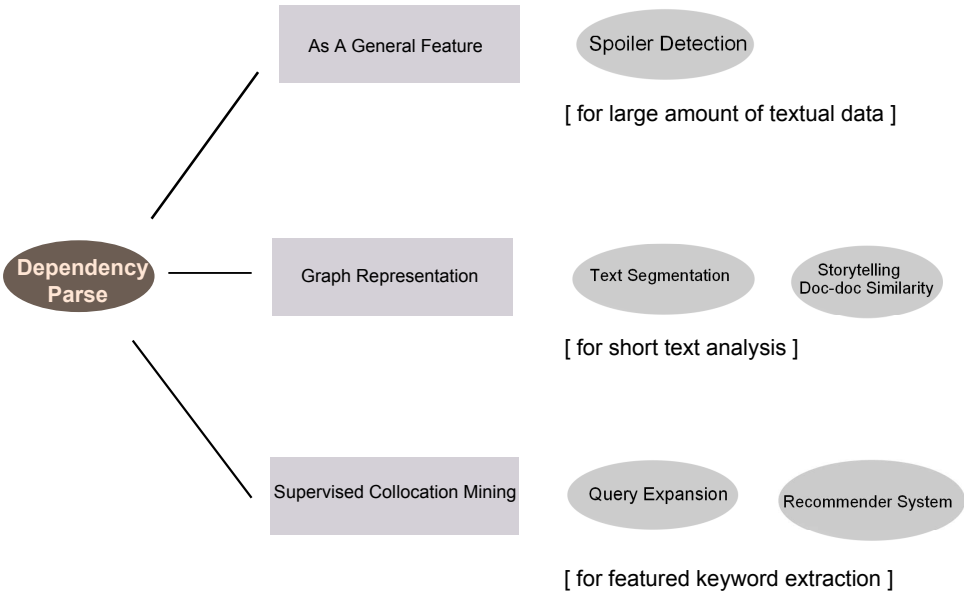


Figure 1.4: Three methods to utilize dependency parses.

Chapter 2

Related Work

We will begin this chapter by surveying methods for representing and reasoning about text.

2.1 Basic Feature Representations for Text

A most popular representation for text is the bag of words (BOW) representation. As shown in Fig. 2.1, preprocessed tokens are treated as the basic units of the text.

Figure 2.1: Bag of words representation of text.

In a BOW model, complex relationships between tokens are ignored (e.g., the order of the words), and occurrence or non-occurrence of a token is simply used. Thus a document D is represented

as a binary vector of occurrences $(w_1, w_2, \ldots, w_n)$ in the simple case. More commonly, frequency and weighting are used in the vector [98],

$$tf \cdot idf = tf_{t,d} \times idf_t$$

where $tf_{t,d}$ denotes the term frequency of t in document d, and idf_t, as the inverse document frequency, denotes the importance of the term in the document collection.

$$idf_t = \log(\frac{N}{df_t})$$

where N is the number of documents in the corpus, and df_t is the number of documents with term t.

As an enriched version of BOW, the n-gram model uses continuous n tokens instead of independent words (thus, the BOW approach can be viewed as using unigrams). N-grams are commonly discussed in the context of language models, where n-grams are depicted using probability distributions [95, 98, 147]. A sample bigram language model looks like

$$p(t_1 t_2 t_3 t_4) = p(t_1)p(t_2|t_1)p(t_3|t_2)p(t_4|t_3)$$

Some works have integrated language models with syntactic structure [11, 25, 48]. However, they focus on the problem of how syntactic relations can help in improving a specific language model, whereas in our work, we focus on dependency parses as a more general feature. Furthermore, in our work, dependency parses are not integrated with any specific language model, nor in a tightly coupled way. We treat dependency parse as a more general text feature for text feature extraction and selection, and we analyze its superiority in many applications in reference to others.

2.2 Text Preprocessing and Feature Selection/Extraction

2.2.1 NLP Tools

Text mining deals with the problem of mining new knowledge from text, mostly free text. In the real world, most text exists in unstructured format. This requires the preprocessing of the free text and as shown in Fig. 1.1, can be conducted in many different levels.

1. **Token level**
   Many Natural Language Processing tools provide the functionality of preprocessing the text at the token level. A tokenizer normally splits the text and returns a series of tokens. A stop list can be used to remove unnecessary stop words such as ‘the’, ‘a’, ‘this’, etc. A stemmer is then typically used to find approximately the root of a word (e.g., we obtain ‘stem’ from ‘stemming’, ‘stemmer’, or ‘stemmed’) [128], while a lemmatizer has a more

complex process and is used to determine the lemma of a word (e.g., we can obtain 'good' from the given word 'better').

Part-of-Speech (POS) tagging [32, 146] is another important technique for text preprocessing. It defines the morphological behavior of a word. Words with different POS tags could be very different in the task of text mining. Named Entity Recognition (NER) [113] is the process of identifying the special categories of words, such as names of people, organizations, or whether it is a location or any other terms fitting into a predefined category. For some special domain, such as biology, researchers have developed special NER tools to identify the domain-specific categories of terms (Genes, proteins, organisms, etc.). Word sense disambiguation (WSD) [34, 114, 140] helps to identify the meaning of a polysemous word when it is located in a specific context.

2. **Syntactic level**
   Some of the other NLP tools used in text preprocessing consider not only the single term, but also the syntactic relationships between terms. Coreference resolution [15, 49, 87] is used to identify the situations when two expressions refer to the same thing. For example, in the sentence 'Bill said he would come this afternoon', 'Bill' and 'he' refer to the same person. Some other special cases of coreference resolution problem also have been studied, e.g., 'other anaphora' [15] aims to find what the word 'other' means and the correct antecedent(s). Other work discusses syntactic features special to technical sublanguage reporting [103].

3. **Semantic level**
   Semantic role labeling (SRL) helps to identify constituents which can fit some general or domain-specific semantic roles [50, 74], e.g., verb-specific roles such as EMPLOYER and EMPLOYEE for the verb *employ*. A SRL system could be very complex, and involves many NLP processing steps. Relation extraction is also an important area in information extraction. Many researchers discuss the extraction of semantic relationships for application domains such as finance or biology (drug-gene relationship, gene-disease relationship, protein-protein interaction, etc.) [46, 135].

Although we give a rough categorization of NLP techniques used in text mining as above, a specific system might straddle multiple categories. For example, a word sense disambiguation algorithm may involve some syntactic analysis, and semantic role labeling may also be used to identify lexical or syntactic information [74]. In general, most preprocessing and feature selection/extraction works for text mining heavily rely on linguistic analysis[1].

2.2.2 Feature Dimensionality Reduction

Real world text data usually is in a high dimensional space. Considering that the number of features in a document is relatively large, dimensionality reduction is widely used in text mining.

[1]Some of these widely used data/text mining tools include: (1) Weka (http://www.cs.waikato.ac.nz/ml/weka/), (2) OpenNLP (http://incubator.apache.org/opennlp/), and (3) LingPipe (http://alias-i.com/lingpipe/).

Dimensionality reduction is necessary not only because it can lead to economical data storage, but also because it can help improve most machine learning algorithms for text mining, in terms of both speed and accuracy. Although generally more features imply higher accuracy, the fact is that too many features make the training process harder and result in the "curse of dimensionality". Dimensionality reduction helps to find the most useful features from thousands of features in a corpus.

A significant amount of work has been expended in this space, some of them are using traditional linear techniques, such as factor analysis [52], Linear Discriminant Analysis [41], and Principal Components Analysis (PCA) [69, 75], others using non-linear methods [138, 166], such as Multidimensional scaling (MDS) [28, 83], Isomap [156], Maximum Variance Unfolding (MVU) [172], Kernel PCA (KPCA) [143], Diffusion Maps (DM) framework [88, 116], Multilayer autoencoders [64], Local Linear Embedding (LLE) [137], Laplacian Eigenmaps [8], Hessian LLE (HLLE) [35], Local Tangent Space Analysis (LTSA) [178], Locally Linear Coordination (LLC) [155], and Manifold charting [20]. Some good review papers about this topic are also available [44, 160].

2.3 Dependency Parsing

Since in this document we focus on utilizing dependency parses as text features, we briefly introduce the related work in dependency parsing and related text mining techniques, such as text preprocessing and feature selection and representation, as well as their applications.

In the field of natural language processing, syntactic parsing has been a mainstay for a significant amount of time. It extracts sentences' grammatical structure according to some predefined grammar. At the same time, syntactic parsing also distills the important components from the plain text. Knowledge from parsing can be widely used in many areas such as information extraction, text mining, information retrieval, and machine translation.

2.3.1 Popular Syntactic Parsers

1. **MiniPar Parser**
   MiniPar [90] is a parser specifically designed for parsing English only. It is a broad-coverage and principle based parser. MiniPar is implemented in C++, and it runs very efficiently[2].

2. **Collins Parser**
   Collins Parser is developed by Michael Collins, and coded in C[3]. Collins Parser is open source software, and requires tagged input text. Dan Bikel's parser extends the Collins parser, and supports multiple languages, such as English, Chinese and Arabic[4] [13].

[2]http://webdocs.cs.ualberta.ca/ lindek/minipar.htm
[3]http://people.csail.mit.edu/mcollins/code.html
[4]http://www.cis.upenn.edu/ dbikel/software.html#stat-parser

3. **Stanford Parser**
   Stanford Parser[5] is developed in Stanford University, and now is still an active project [102]. It is implemented in Java, and supports English, Chinese, German, and Arabic. It also can be used for other languages, such as Italian, Bulgarian, and Portuguese. The Stanford Parser is also open source software, and is widely used in academic research work, though it is not as fast as some other parsers.

4. **Berkeley Parser**
   Berkeley Parser[6] [123, 124] learns probabilistic context-free grammars (PCFGs) to assign sentences with the most likely grammatical structure. Its source code is also freely available, and supports multiple languages such as Chinese and German.

5. **OpenNLP**
   OpenNLP[7] is an incubator project from Apache. It is a group of open source natural language processing sub-projects. It performs most of the NLP tasks such as sentence detection, tokenization, part-of-speech tagging, name entity detection, sentence parsing, etc. OpenNLP parser reads in tokenized English sentences and the pre-trained models, then outputs the parse trees. Meanwhile, we can train our own model according to various corpora.

6. **Others**
   Some other complicated text systems also provide similar functionality for parsing sentences. GATE[8] (General Architecture for Text Engineering) provides an interface for plugging in many different kinds of NLP tools. Apache UIMA[9] (Unstructured Information Management Architecture), as an open source implementation of the UIMA specification, also comes with many NLP components wrappers, such as OpenNLP parser wrapper.

2.3.2 Applications

Parsing has many applications. In statistical machine translation (SMT), resolving the syntactic structure of a sentence helps mapping components in two sentences written in different languages. A good example is that in English a sentence is written as "*subject* $\longrightarrow$ *verb* $\longrightarrow$ *object*", while in Japanese, a sentence has the structure of "*subject* $\longrightarrow$ *object* $\longrightarrow$ *verb*". Mapping those components in a training corpus can improve research work in machine translation. In [131], a dependency parser is applied on the source language, and a word segmentation/alignment component is applied on the targeting language. Combined with conventional SMT models, dependency tree based techniques show promising performance.

[5] http://www-nlp.stanford.edu/software/lex-parser.shtml
[6] http://code.google.com/p/berkeleyparser/
[7] http://incubator.apache.org/opennlp/
[8] http://gate.ac.uk/gate/doc/plugins.html
[9] http://uima.apache.org/

Other potential applications could be for information extraction, question answering, etc. In information extraction, utilizing a parsing tree can help to understand the text in a better way. In [31], a dependency tree kernel is used for relation extraction, and it shows a vast improvement over the classic bag-of-words kernel. In question answering, syntactic structure can be used to serve as a feature to improve the classifier in a conventional way. In [167], syntactic structure is used to loosely connect the question and the correct answers.

2.3.3 Challenges for Current Parsers

1. **Ambiguity**

   The ambiguity characteristics of language bring a lot of challenges to parsing tasks. In many cases, sentences can be interpreted in different ways. The lexical ambiguity before parsing could harm the accuracy of the syntactic analysis. For example, "requests" could be a noun or a verb. Incorrect part-of-speech tagging makes the follow-up syntactic parsing difficult. Syntactic ambiguity makes it even worse. In the sentence

   "*The company in this small town along I-81 is new.*"

   "*along I-81*" could be used to describe the small town, or the company, which is unclear without more contextual information. When multiple prepositional phrases are attached in a sentence, this problem becomes very common, and sometimes it is hard to get the correct grammatical structures. Another similar example is

   "*The boy behind the old man in the room is crying*".

   One of the other types of ambiguity is called coordination ambiguity:

   "*Books on the table and laptop are mine.*"

   It is not easy to figure out whether "laptop is mine" or "books on the laptop are mine".

   Besides the prepositional phrase attachment ambiguity and the coordination ambiguity mentioned above, some additional examples which make syntactic parsing difficult are as follows:

   "*They are hunting dogs.*"
   "*Flying planes can be dangerous.*"
   "*I saw the guy with a telescope.*"

2. **Performance**
   Today, parsing sentences is still relatively slower compared to other NLP processing steps.

For example, when dealing with a long sentence, the Stanford Parser may take several seconds, which makes it very hard to handle large scale text corpora. However, with the development of cloud computing, this performance issue may be alleviated. The MapReduce computing framework and key-value storage make data analysis easier in the large scale. For example, Hadoop[10] and HDFS[11] are now widely used in many companies dealing with "big data", such as Facebook, Twitter, and LinkedIn. Although these systems were originally designed to solve the problem of data intensive tasks, the convenience of the Map/Reduce framework makes it easier for handling data analysis jobs. Apache Mahout[12] is such an open source library that makes machine learning scalable to reasonably large size of data. Even in academic institutions where building such a large distributed system is infeasible, cloud computing facilities (such as Amazon's EC2 and S3) make research practical by enabling the rental of clusters.

2.4 Related Work in Text Mining Applications

We now focus on work related to the specific tasks studied in this dissertation research.

2.4.1 Text Similarity

Measuring semantic text similarity has been investigated for a long time; [68] gives a very good survey about different methods of calculating semantic text similarity. Four major categories of semantic text similarity are listed:

1. **Vector-based**
   The vector space model is widely used in information retrieval. It represents text as a vector of terms, and text similarity is measured as the word vector similarity. The vector space model assumes that semantically related documents usually share many words. This may cause many problems when the two text snippets are small, since the vector dimension is normally very large. Meanwhile, synonymy and polysemy bring more disambiguation into the vector. In the case of synonymy, vectors with different terms may still be semantically related while for polysemy, those same words in different vectors could refer to two different things.

2. **Corpus-based**
   The corpus based method heavily relies on information content derived from the corpus. Latent Semantic Analysis (LSA) [85] can be used to analyze large scale corpora and extract the

[10] http://hadoop.apache.org/
[11] http://hadoop.apache.org/hdfs/
[12] http://mahout.apache.org/

hidden term-term semantic connections. Singular Value Decomposition (SVD) is used and text similarity is measured by the similarity between two vectors in a reduced dimensionality space. Hyperspace Analogues to Language (HAL) [21] is another method which utilizes the lexical co-occurrence to construct the word-word semantic space.

3. **Hybrid method**
   Hybrid method can be treated as a combination of corpus-based method and knowledge-based (dictionary/thesaurus-based) method. The knowledge-based method uses external resources, such as semantic networks, dictionaries (e.g., WordNet), to quantify the semantic connection strength between words. [109] presents a good example of a hybrid method.

4. **Feature-based**
   Feature-based method considers each text as a feature set, and uses machine learning methods to capture the similarity value between them.

2.4.2 Query Expansion

Query expansion is the process of reformulating or expanding a seed query to improve information retrieval accuracy. It has been treated as an important research topic for search engines for a very long time and has been thoroughly investigated. In general, it may fit into the following categories:

1. **Global/Local analysis**
   By analyzing large scale text collections, term-term co-occurrence information can be extracted for query expansion in a global view [4, 72, 130]. Meanwhile, local analysis uses only top-ranked documents after pseudo-relevance feedback [110, 117, 175, 176].

2. **Log-based**
   Search logs also provide very helpful information to expand queries. Term-term relationships can be constructed by analyzing the log file of past query usage [14, 30, 45, 168].

3. **Ontology-based**
   Ontology is well-organized and predefined information, such as WordNet[13]. It is natural to utilize ontology to enhance query expansion according to the term-term relationship in the ontology system. A review of ontology-based query expansion is given in [12].

4. **Domain-specific**
   For some cases, traditional query expansion methods may not fit very well. Given a seed query, simply using the term-term relationship or co-occurrence information cannot generate high-quality expanding query terms. These cases fall into the same category of domain/scenario-specific query expansion as in [94, 43, 118]. Different from the general web search, domain-specific query expansion requires a domain-relevant expanding query term instead of one

[13] http://wordnet.princeton.edu

related to the original term. [94] considers the typical scenario of searching treatment for a given disease term from a medical text corpus. Obviously the traditional query expansion is not suitable since expanding terms based on the correlation with the initial disease term may not help for the treatment scenario (e.g., the traditional query expansion will generate *lung excision* for the original query of *lung cancer*, but this will not help for the treatment searching scenario). [43] uses SVM to generate the query expansion for the home page search. [118] uses the decision tree to extract the keyword "spices" as query expansion from both the domain-relevant and irrelevant training documents. All these domain-specific approaches such as [43, 94, 118] focus on different application level needs.

2.4.3 Recommender Systems

Recommender systems are designed to give suggestions to end users. They can be mainly divided into two types: content-based and collaborative filtering.

Content-based filtering recommends items similar to those the user has liked/rated before. It collects the information from the user's history data, models the user's interest, and predicts new items that the user may like. Normally, the model has to be studied by analyzing both the historical content data and the user's profile information. The most obvious advantage of content-based method is that it is not highly affected by the "new item" problem, while in collaborative filtering, items which have not been rated before have no historical data, and it is hard for the system to recommend them. However, content-based method is not good when dealing with the "new user" problem where there is not enough historical data for a user to model his/her interest.

Besides the content-based filtering method, collaborative filtering has been studied for a very long time. For different application areas, many systems, such as memory-based, model-based, and hybrid recommender systems have been proposed [150]. Item-based filtering systems [91, 136] focus on alleviating the data sparsity problem of memory-based systems. As a prevalent algorithm, the item-based method is easy to implement. It first calculates the item similarity matrix, and then it predicts the rating values for item i by considering the user's history ratings and the relationship between i and the rated items. The top-N recommendation can also be carried out after the item similarity matrix is calculated [33]. The item similarity matrix is mainly calculated by checking how different users rate the same two items. Many different methods have been discussed, such as Pearson correlation and its variants, as well as cosine similarity and its variants [136].

2.5 Outline of Document

The structure of this document is described below.

Chapter 3 presents an application which uses dependency parses to help improve information extraction, and uses them in conjunction with topic modeling for automated 'spoiler' detection in

movie reviews.

Chapter 4 presents constructing document similarity by using dependency parses and its application in text segmentation.

Chapter 5 presents an application of dependency parses in query expansion, specifically in drug interaction search. Dependency parses are used to formulate long-span collocations, and statistical measures are used to rank query candidates using these collocations.

Chapter 6 presents the application to recommender systems.

Chapter 3

Automatic Spoiler Tagging using Linguistic Cues

Given a movie comment, does it contain a spoiler? A spoiler is a comment that, when disclosed, would ruin a surprise or reveal an important plot detail. We study automatic methods to detect comments and reviews that contain spoilers and apply them to reviews from the IMDb (Internet Movie Database) website. We develop topic models, based on Latent Dirichlet Allocation (LDA), but using linguistic dependency information in place of simple features from bag of words (BOW) representations. Experimental results demonstrate the effectiveness of our technique over four movie-comment datasets of different scales.

3.1 Introduction

In everyday parlance, the notion of 'spoilers' refers to information, such as a movie plot, whose advance revelation destroys the enjoyment of the consumer. For instance, consider the movie *Derailed* which features Clive Owen and Jennifer Aniston. In the script, Owen is married and meets Aniston on a train during his daily commute to work. The two of them begin an affair. The adultery is noticed by some unscrupulous people who proceed to blackmail Owen and Aniston. To experience a spoiler, consider this comment from *imdb.com*:

> I can understand why Aniston wanted to do this role, since she gets to play majorly against type (as the supposedly 'nice' girl who's really - oh no! - part of the scam), but I'm at a loss to figure out what Clive Owen is doing in this sub-par, unoriginal, ugly and overly violent excuse for a thriller.

i.e., we learn that Aniston's character is actually a not-so-nice person who woos married men for later blackmail, and thus a very suspenseful piece of information is revealed. Automatic ways to detect spoilers are crucial in large sites that host reviews and opinions.

Arguably, what constitutes a spoiler is inherently a subjective assessment and, for movies/books with intricate storylines, some comments are likely to contain more spoilers than others. We therefore cast the spoiler detection problem as a ranking problem so that comments that are more likely to be spoilers are to be ranked higher than others. In particular, we rank user comments w.r.t. (i.e., given) the movie's synopsis which, according to *imdb*, is '[a detailed description of the movie, including spoilers, so that users who haven't seen a movie can read anything about the title]'.

Our contributions are three fold. (i) We formulate the novel task of spoiler detection in reviews and cast it as ranking user comments against a synopsis. We demonstrate how simple bag-of-words (BOW) representations need to be augmented with linguistic cues in order to satisfactorily detect spoilers. (ii) We showcase the ability of dependency parses to extract discriminatory linguistic cues that can distinguish spoilers from non-spoilers. We utilize an LDA-based model [171] to probabilistically rank spoilers. Our approach does not require manual tagging of positive and negative examples – an advantage that is crucial to large scale implementation. (iii) We conduct a detailed experimental evaluation with *imdb* to assess the effectiveness of our framework. Using manually tagged comments for four diverse movies and suitably configured design choices, we evaluate a total of 12 ranking strategies.

3.2 LDA

Probabilistic topic modeling has attracted significant attention with techniques such as probabilistic latent semantic analysis (PLSA) [65] and LDA [18, 53, 61, 148]. We discuss LDA in detail due to its centrality to our proposed techniques. As a generative model, LDA describes how text could be generated from a latent set of variables denoting topics. Each document is modeled as a mixture of topics, and topics are modeled as multinomial distributions on words.

An unlabeled training corpus can be used to estimate an LDA model. Many inference methods have been proposed, e.g., variational methods [18], expectation propagation [53], Gibbs sampling [53], and a collapsed variational Bayesian inference method [154]. Gibbs sampling, as a specific form of Markov chain Monte Carlo (MCMC), is a popular method for estimating LDA models. After an LDA model is estimated, it can be used in a very versatile manner: to analyze new documents, for inference tasks, or for retrieval/comparison functions. For instance, we can calculate the probability that a given word appears in a document conditioned on other words. Furthermore, two kinds of similarities can be assessed: between documents and between words [148]. The similarity between two documents also can be used to retrieve documents relevant to a query document [61]. Yet another application is to use LDA as a dimensionality reduction tool for text classification [18].

To improve LDA's expressiveness, we can relax the bag-of-words assumption and plug in more sophisticated topic models [54, 55, 164, 163, 169, 170]. sLDA (supervised LDA), as a statistical model of labeled collections, focuses on the prediction problem [12]. The correlated topic model (CTM) [17] addresses plain LDA's inability to model topic correlation. The author-topic model (AT) [149] considers not only topics but also authors of the documents, and models documents as

if they were generated by a two-stage stochastic process.

3.3 LDA-based Spoiler Ranking

3.3.1 Methods

Based on the fact that a spoiler should be topically close to the synopsis, we propose three methods to solve the spoiler ranking problem. The first two use LDA as a preprocessing stage, whereas the third requires positive training data.

Predictive perplexity: Our first method is motivated by the use of LDA-based predictive perplexity (PP) for collaborative filtering [18]. Here, the PP metric is evaluated over a fixed test dataset in order to empirically compare LDA with other models (pLSI, mixture of unigrams). In our work, we view documents as analogous to users, and words inside documents as analogous to movies. Given a group of known words, we predict the other group of unknown words. We can either calculate the predictive perplexity value from each movie comment Com to the unique synopsis (PP1), or from the synopsis Syn to each comment (PP2).

$$PP1(Syn, \mathbf{w}_{com}) = exp\{-\frac{\sum_{d=1}^{M_{syn}} \log p(w_d|\mathbf{w}_{com})}{M_{syn}}\}$$

$$PP2(Com, \mathbf{w}_{syn}) = exp\{-\frac{\sum_{d=1}^{M_{com}} \log p(w_d|\mathbf{w}_{syn})}{M_{com}}\}$$

In the equations above, $p(w_d|\mathbf{w}_{com})$ and $p(w_d|\mathbf{w}_{syn})$ are the probabilities to generate the word (w_d) from a group of observed words $\mathbf{w}_{obs}$ (either a comment $\mathbf{w}_{com}$ or a synopsis $\mathbf{w}_{syn}$).

$$p(w|\mathbf{w}_{obs}) = \int \sum_z p(w|z)p(z|\theta)p(\theta|\mathbf{w}_{obs})d\theta$$

M_{com} or M_{syn} is the length of a comment or a synopsis. Notice that $p(\theta|\mathbf{w}_{obs})$ can be easily calculated after estimating the LDA model by Gibbs sampling. It is also mentioned as "predictive likelihood ranking" in [61].

Symmetrized KL-divergence: Since documents are modeled as mixtures of topics in LDA, we can calculate the similarity between synopsis and comment by measuring their topic distributions' similarity. We adopt the widely-used symmetrized Kullback Leibler (KL) divergence [61, 148] to measure the difference between the two documents' topic distributions:

$$sKL(Syn, Com) = \frac{1}{2}[D_{KL}(Syn\|Com) + D_{KL}(Com\|Syn)]$$

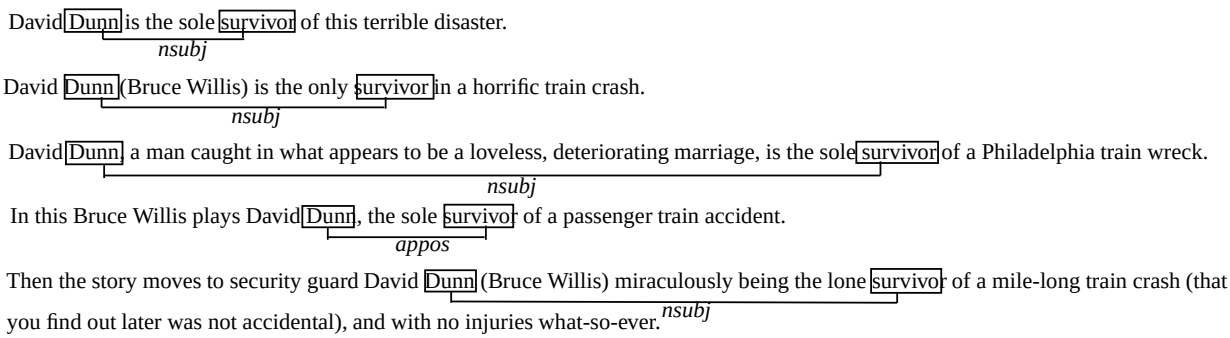


Figure 3.1: Four sentences with the same topical connection between "Dunn" and "survivor". We integrate this relation into LDA by treating it as a virtual word "Dunn-survivor."

where

$$D_{KL}(p\|q) = \sum_{j=1}^{T} p_j \log_2 \frac{p_j}{q_j}$$

LPU: Viewing the spoiler ranking problem as a retrieval task given the (long) query synopsis, we also consider the LPU (Learning from Positive and Unlabeled Data) method [92]. We apply LPU as if the comment collection was the unlabeled dataset, and the synopsis together with a few obvious spoiler comments as the positive training data.

3.3.2 Dependency Parsing

LDA, as a topic model, is widely used as a clustering method and dimensionality reduction tool. It models text as a mixture of topics. However, topics extracted by LDA are not necessarily the same topics as judged by humans since the definition of topic is very subjective. For instance, when conducting sentiment polarity analysis, we hope that topics are clusters concerning one certain kind of subjective sentiment. But for other purposes, we may desire topics focusing on broad 'plots.' Since LDA merely processes a collection according to the statistical distribution of words, its results might not fit either of these two cases mentioned above.

In a basic topic model (section 3.3.1), neither the order of a sequence of words nor the semantic connections between two words affect the probabilistic modeling. Documents are generated only based on a BOW assumption. However, word order information is very important for most text-related tasks, and simply discarding the order information is inappropriate. Significant work has been done to address this problem. Griffiths et al. use order information by incorporating collocations [54, 55]. They give an example of the collocation "*united kingdom*", which is ideally treated as a single chunk rather than two independent words. However, this model can only be used to capture collocations involving sequential terms. Their extended model [55] integrates topics and

syntax, and identifies syntactic classes of words based on their distribution. More sophisticated models exist [164, 169, 170, 163] but all of them are focused on solving linguistic analysis tasks using topic models. In this work, however, our focus is on utilizing dependency information as a preprocessing step to help improve the accuracy of LDA models.

In more detail, we utilize dependency parsing to break down sentences and treat parses as independent 'virtual words,' to be added to the original BOW-based LDA model. In our experiments we employ the Stanford typed dependency parser [1] [102] as our parsing tool. We use collapsed typed dependencies (a.k.a. grammatical relations) to form the virtual words. However, we do not incorporate all the dependencies. We only retain dependencies whose terms have part-of-speech tags such as "*NN*", "*VB*", "*JJ*", "*PRP*" and "*RB*"[2], since these terms have strong plot meaning, and are close to the movie topic. Fig. 3.2 shows a typical parsing result from one sample sentence. This sentence is taken from a review of *Unbreakable*.

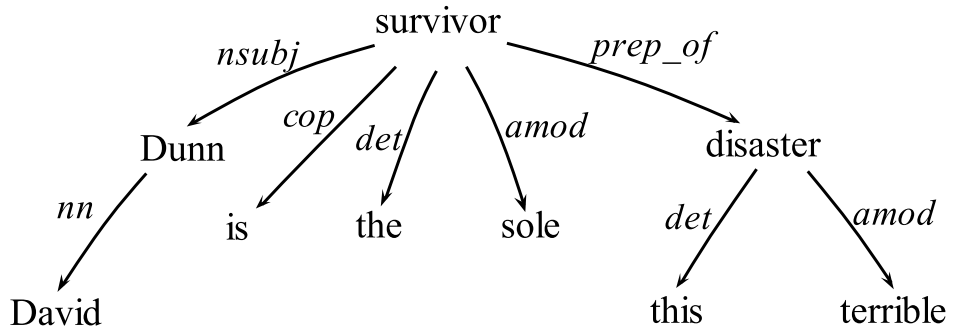


Figure 3.2: Dependency parse of "David Dunn is the sole survivor of this terrible disaster".

Consider Fig. 3.1, which depicts five sample sentences all containing two words: "*Dunn*" and "*survivor*". Although these sentences appear different, these two words above refer to the same individual. By treating dependencies as virtual words, we can easily integrate these plot-related relations into an LDA model. Notice that among these five sentences, the grammatical relations between these two words are different: in the fourth sentence, "*survivor*" serves as an appositional modifier of the term "*Dunn*" (appos), whereas in other sentences, "*Dunn*" serves as the nominal subject of "*survivor*" (nsubj). What is important to note is that the surface distances between these given words in different sentences vary a lot. By utilizing dependency parsing, we can capture the semantic connection which is physically separated by even as much as 15 words, as in the third sentence.

We evaluate *topic drift* among the results from plain LDA. We mainly check whether plain LDA will assign the same topic to those terms that have specific linguistic dependency relations. We

[1] http://nlp.stanford.edu/software, V1.6

[2] In the implementation, we actually considered all the POS tags with these five tags as prefix, such as "*NNS*", "*VBN*", etc.

only consider the following four types of dependencies for evaluation[3]:

- Relations with two noun terms: <NN, NN>, such as "*appos*", "*nn*", "*abbrev*" etc.;
- Relations with one noun and one adjective: <NN, JJ>, like "*amod*";
- Relations with one noun and one verb: <NN, VB>, such as "*agent*", "*dobj*", etc.;
- Relations with only one noun: <NN, *>, which is the relaxed version of <NN, NN>;

We experimented with different pre-set topic numbers (500, 50, and 2) and conducted experiments on four different movie comment collections with LDA analysis. Table 1 shows that <NN, NN> dependency has the highest chance to be topic-matched[4] than other relations. However, all dependencies have very low percentage to be topic-matched, and with a topic number of 2, there remained a significant amount of unmatched <NN, NN> dependencies, demonstrating that simply doing plain LDA may not capture the plot "topic" as we desire.

Observing the results above, each method from section 3.3.1 (PP1, PP2, sKL and LPU) can be extended by: (1) using BOW-based words, (2) using only dependency-based words, or (3) using a mix of BOW and dependency (dependencies as virtual words). This induces 12 different ranking strategies.

3.4 Experimental Results

3.4.1 Data Preparation

IMDb boasts a collection of more than 203,000 movies (from 1999 to 2009), and the number of comments and reviews for these movies number nearly 970,000. For those movies with synopsis provided by IMDb, the average length of their synopses is about 2422 characters[5]. Our experimental setup, for evaluation purposes, requires some amount of labeled data. We choose four movies from IMDb, together with 2148 comments. As we can see in Table 3.3, these four movies have different sizes of comment sets: the movie "Unbreakable" (2000) has more than 1000 comments, whereas the movie "Role Models" (2008) has only 123 comments.

We labeled all the 2148 comments for these four movies manually, and as Table 3.3 shows, about 20% of each movie's comments are spoilers. Our labeling result is a little different from the current labeling in IMDb: among the 2148 comments, although 1659 comments have the same labels with IMDb, the other 489 are different (205 are treated as spoilers by IMDb but non-spoilers by us;

[3]Here we use <NN, JJ> to express relations having NN and JJ terms, but not necessarily in that order. Also, NN represents all tags related with nouns in the Penn Treebank Tagset, such as NNS. This applies to all the four expressions here.

[4]When both the left term and the right term of a dependency share the same topic, the relation is topic-matched.

[5]Those movies without synopsis are not included.

Table 3.1: Topic match analysis for plain LDA (Each entry is the ratio of topic-matched dependencies to all dependencies)

topic number = 500				
Movie Name	<NN, NN>	<NN, JJ>	<NN, VB>	<NN, *>
Unbreakable	772/3024	412/4411	870/19498	5672/61251
Blood Diamond	441/1775	83/553	80/1012	609/3496
Shooter	242/1846	42/1098	114/2150	1237/15793
Role Models	409/2978	60/1396	76/2529	559/7276
topic number = 50				
Movie Name	<NN, NN>	<NN, JJ>	<NN, VB>	<NN, *>
Unbreakable	1326/3024	953/4411	3354/19498	14067/61251
Blood Diamond	806/1775	151/553	210/1012	1194/3496
Shooter	584/1846	204/1098	392/2150	3435/15793
Role Models	1156/2978	190/1396	309/2529	1702/7276
topic number = 2				
Movie Name	<NN, NN>	<NN, JJ>	<NN, VB>	<NN, *>
Unbreakable	2379/3024	3106/4411	13606/19498	43876/61251
Blood Diamond	1391/1775	404/553	761/1012	2668/3496
Shooter	1403/1846	768/1098	1485/2150	11008/15793
Role Models	2185/2978	908/1396	1573/2529	4920/7276

Table 3.2: Some examples of incorrect spoiler tagging in IMDb (sentences in italics are spoiler comments).

No.	Tag by IMDb	Comment in IMDb
1	Spoiler	The whole film is somewhat slow and it would've been possible to add more action scenes. Even though I liked it very much (6.8/10) I think it is less impressive than "The Sixth Sense" (8.0/10). I would like to be more specific with each scene but it will turn this comment into a spoiler so I will leave it there. I recommend you to see the movie if you come from the basic Sci-Fi generation, otherwise you may feel uncomfortable with it. Anyway once upon a time you were a kid in wonderland and everything was possible. [tt0217869]
2	Spoiler	This is one of the rare masterpiece that never got the respect it deserved because people were expecting sixth sense part 2.Sixth sense was a great film but this is M.N. Shyamalan's best work till date. This is easily one of my top 10 films of all time. Excellent acting, direction, score, cinematography and mood. This movie will hold you in awe from start to finish and any student of cinema would tell what a piece of art this film is. The cast is phenomenal, right from bruce willis to sam jackson and penn , everyone is spectacular in their roles and they make u realise that you do not need loud dramatic moments to create an impact, going slow and subtle is the trick here. This is not a thriller, it's a realistic superhero film. [tt0217869]
3	Spoiler	I can't believe this movie gets a higher rating than the village. OK, after thinking about it, i get the story of unbreakable and i understand what it's trying to say. I do think the plot and the idea is captivating and interesting. Having said that, i don't think the director did anything to make this movie captivating nor interesting. It seemed to try too hard to make this movie a riddle for the audience to solve. The pace was slow at the beginning and ended just as it was getting faster. I remember going out of the cinema, feeling frustrated and confused. it's not until i thoroughly thought about it that i understood the plot. I believe a good movie should engaged the audience and be cleverly suspenseful without confusing the audience too much. Unbreakable tried to be that but failed miserably. 2 out of 10, see the village instead. [tt0217869]
4	Spoiler	This movie touched me in ways I have trouble expressing, and brings forth a message one truly need to take seriously! I was moved, and the ending brought a tear to my eye, as well as a constant two-minute shiver down my spine. It shows how our western way of life influence the lives of thousands of innocents, in a not-so-positive way. Conflict diamonds, as theme this movie debates, are just one of them. Think of Nike, oil, and so on. We continually exploit "lesser developed" nations for our own benefit, leaving a trail of destruction, sorrow, and broken backs in our trail. I, for one, will be more attentive as to what products I purchase in the future, that's for sure. [tt0450259]
5	Non-spoiler	... But the movie takes a while to get to the point. ***"Mr. Glass" has caused lots of mass tragedies in order to find the UNBREAKABLE person. Thus, he is both a mentor and a MONSTER***. ... [tt0217869]
6	Non-spoiler	... This film is about a sniper who loses his best friend while on a shooting mission. A few years later, he is now retired and living in a woodland with his do. Then he is visited by the military to plan an assassination of the president. The shot is fired. ***Unfortunately he is set up to being the shooter and is hunted by cops everywhere. He must find out why he has been set up and also try and stop the real killers***. ... [tt0822854]

vice versa with 284) The current labeling system in IMDb is very coarse: as shown in Table 3.2, the first four rows of comments are labeled as spoilers by IMDb, but actually they are not (movie IDs are appended at the end of each row). The last two rows of comments are ignored by IMDb; however, they do expose the plots about the twisting ends.

After crawling all the comments of these four movies, we performed sentence chunking using the LingPipe toolkit and obtained 356 sentences for the four movies' synopses, and 26964 sentences for all the comments of these four movies. These sentences were parsed to extract dependency information: we obtained 5655 dependencies for all synopsis sentences and 448170 dependencies for all comment sentences. From these, we only retain those dependencies that have at least one noun term in either the left side or the right side. For measures which require dependency information, the dependencies are re-organized and treated as a new term planted in the text.

Table 3.3: Evaluation dataset about four movies with different numbers of comments.

Movie Name	IMDb ID	#Comments	#Spoilers
Unbreakable	tt0217869	1219	205
Blood Diamond	tt0450259	538	147
Shooter	tt0822854	268	73
Role Models	tt0430922	123	39

3.4.2 Experiments

- **Topic number analysis**

One of the shortcomings of LDA-based methods is that they require setting a number of topics in advance. Numerous ways have been proposed to handle this problem [16, 18, 53, 55, 61, 148, 153]. Perplexity, which is widely used in the language modeling community, is also used to predict the best number of topics. It is a measure of how well the model fits the unseen documents, and is calculated as average per-word held-out likelihood. The lower the perplexity is, the better the model is, and therefore, the number of topics is specified as the one leading to the best performance. Griffiths and Steyvers [53] also discuss the standard Bayesian method which computes the posterior probability of different models given the observed data. Another method from non-parametric Bayesian statistics automatically helps choose the appropriate number of topics, with flexibility to still choose hyper-parameters [16, 153]. Although the debate of choosing an appropriate number of topics continues [19], we utilized the classic perplexity method in our work. Heinrich [61] demonstrated that perplexity can be calculated by:

$$P(\tilde{\mathcal{W}}|\mathcal{M}) = \prod_{m=1}^{M} p(\tilde{\vec{w}}_m|\mathcal{M})^{-\frac{1}{N}} = exp\{-\frac{\sum_{m=1}^{M} \log p(\tilde{\vec{w}}_m|\mathcal{M})}{\sum_{m=1}^{M} N_m}\}$$

We chose different topic numbers and calculated the perplexity value for the 20% held-out comments. A good number of topics was found to be between 200 and 600 for both BOW-based strategy and BOW+Dependency strategy, and is also affected by the size of movie comment collections. (We used 0.1 as the document topic prior, and 0.01 as the topic word prior.)

- **LDA analysis process**

As discussed earlier, our task is to rank all the comments according to their possibilities of being a spoiler. We primarily used four methods to do the ranking: PP1, PP2, sKL, and the LPU method. For each method, we tried the basic model using "bag-of-words", and the model using dependency parse information (only), and also with both BOW and dependency information mixed. We utilize the LingPipe LDA clustering component which uses Gibbs sampling.

Among the four methods studied here, PP1, PP2 and sKL are based on LDA preprocessing. After obtaining the topic-word distribution and the posterior distributions for topics in each document, the PP1 and PP2 metrics can be calculated easily. The symmetrized KL divergence between each pair of synopsis and comment is calculated by comparing their topic distributions. LPU method, as a text classifier, requires a set of positive training data. We selected those comments which contain terms or phrases as strong hint of spoiler (using a list of 20 phrases as the filter, such as "spoiler alert", "spoiler ahead", etc.). These spoiler comments together with the synopsis, are treated as the positive training data. We then utilized LPU to label each comment with a real number for ranking.

3.4.3 Evaluation

To evaluate the ranking effects of the 12 strategies, we plot *n*-best precision and recall graphs, which are widely used for assessing collocation measures [38, 122]. Fig. 3.3 shows the precision-recall graphs from 12 different measures for the four movie comment collections. The *x*-axis represents the proportion of the ranking list, while the *y*-axis depicts the corresponding precision or recall value. The upper part of the figure is the result for the movie which contains more than 1000 comments, while the bottom part demonstrates the result for the relatively small comment collection. The *n*-best evaluation shows that for all the four movie comment collections, PP1_mix and PP1 perform significantly better than the other methods, and the dependency information helps to increase the accuracy significantly, especially for the larger size collection. The LPU method, though using part of the positive training data, did not perform very well. The reason could be that although some of the users put the warning phrases (like "spoiler alert") ahead of their comments, the comment might contain only indirect plot-revealing information. This also reflects that a spoiler tagging method by using only keywords typically will not work. Finally, the approach to directly calculate the symmetrized KL divergence seems to be not suitable, either.

3.4.4 LDA Iteration Analysis

We also compared the *average precision* values and *normalized discounted cumulative gain* (nDCG) values [29, 70] of the ranking results with different parameters for Gibbs sampling, such as burnin period and sample size. Average precision is calculated by averaging the precision values from the ranking positions where a valid spoiler is found, and the nDCG value for the top-p list is calculated as $nDCG_p = \frac{DCG_p}{IDCG}$, where DCG_p is defined as:

$$DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i}$$

where rel_i is 1 when the *i*-th comment in the list is judged as a real spoiler, and 0, otherwise. IDCG denotes the maximum possible DCG value when all the real spoilers are ranked at the top (*perfect ranking*) [70].

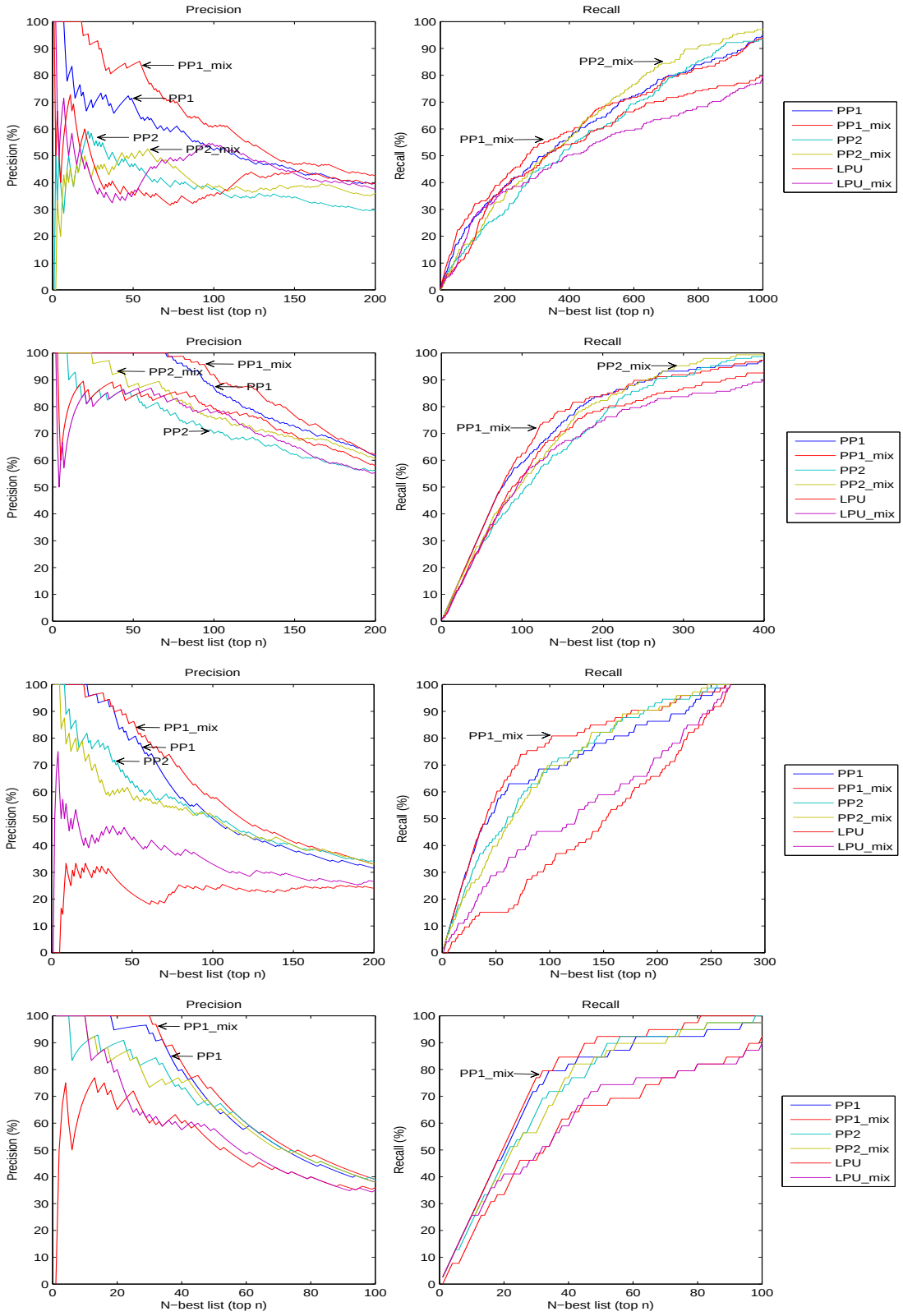


Figure 3.3: N-best (top *n*) evaluation (Burnin period = 100): comparison of precision-recall for different methods on four movie comment collections. The PP1 method with BOW and dependency information mixed performs the best among all the measures. Other six methods such as dependency only and KL-based which do not give good performance are ignored in this figure to make it readable. Full comparison information is available at: http://sites.google.com/site/ldaspoiler/

Table 3.4: Comparison of ranking by PP_mix using different parameters for Gibbs sampling (analyzed on the top 150 ranking lists, and the values in the table are the mean of the accuracy from four movie comment collections).

Burnin	<S=100; Lag=2>		<S=10; Lag=2>		<S=1; Lag=2>	
	AvgP (%)	nDCG	AvgP (%)	nDCG	AvgP (%)	nDCG
400	80.85	0.951	78.2	0.938	78.1	0.94
200	80.95	0.951	80.5	0.948	79.1	0.94
100	87.25	0.974	80.2	0.943	82.4	0.96
50	81.5	0.958	79.5	0.942	80.0	0.94
10	78.9	0.944	79.5	0.949	75.9	0.92
1	79.4	0.940	79.2	0.952	58.0	0.86

As we can see from Table 3.4, the accuracy is not affected too much as long as the burnin period for the MCMC process is longer than 50 and the sample size retained is larger than 10. In our experiments, we use 100 as the burnin parameter, and beyond that, 100 samples were retained with sample lag of 2.

3.4.5 Representative Results

As shown in Table 3.5, we find that the basic BOW strategy prefers the longer comments whereas the strategy that uses dependency information prefers the shorter ones. Although it is reasonable that a longer comment would have a higher probability of revealing the plot, methods which prefer the longer comments usually leave out the short spoiler comments. By incorporating the dependency information together with the basic BOW, the new method reduces this shortcoming. For instance, consider one short comment for the movie "*Unbreakable* (2000)":

> This is the same formula as Sixth Sense – from the ability to see things other people don't, to the shocking ending. Only this movie is just not plausible – I mean Elijah goes around causing disasters, trying to see if anyone is "Unbreakable" – it's gonna take a lot of disasters because its a big world.

which is ranked as the 27th result in the PP1_mix method, whereas the BOW based PP1 method places it at the 398th result in the list. Obviously, this comment reveals the twisting end that it is Elijah who caused the disasters.

Table 3.5: Comparison of average length of the top50 comments of 4 movies from 2 strategies.

	Role Models	Shooter	Blood Diamond	Unbreakable
BOW	2162.14	2259.36	2829.86	1389.18
Dependency	1596.14	1232.12	2435.58	1295.72

3.5 Conclusions and Future Work

We have introduced the spoiler detection problem and proposed using topic models to rank movie comments according to the extent they reveal the movie's plot. In particular, integrating linguistic cues from dependency information into our topic model significantly improves the ranking accuracy.

In future work, we seek to study schemes which can segment comments to potentially identify the relevant spoiler portion automatically. The automatic labeling idea of [107] also can be studied in our framework. Deeper linguistic analysis, such as named entity recognition and semantic role labeling, also can be conducted. In addition, evaluating topic models or choosing the right number of topics using dependency information can be further studied. Finally, integrating the dependency relationships more directly into the probabilistic graphical model is also worthy of study.

Chapter 4

Dependency Parses for Document Similarity and Segmentation

In this chapter, we study the use of dependency parses for capturing document similarity and how such a similarity can be used for segmenting text.

4.1 Introduction

Metrics for capturing document similarity have been well studied, e.g.:

$$Cos(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

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

$$Dice(A, B) = \frac{2|A \cap B|}{|A| + |B|}$$

The cosine similarity measure treats documents as two vectors, and captures the angle between the two vectors. The Jaccard and Dice coefficients typically treat documents as sets (bags) of words, and capture the similarity and overlap between these sets, although vector-space versions of these coefficients exist. All these methods above are commonly used with the bag-of-words model. When vector-space representations are used, a common weighting scheme is tf-idf.

In the absence of specific domain knowledge, it is difficult to ascertain which of the above measures are the most suitable for a given application. A dependency parse can be viewed as an alternate representation of documents so that parses from each sentence can be merged together to form a

document level graph, not unlike a concept map. With richer information arising from the dependency analysis, we can specifically focus on certain types of dependencies, and thus filter out the non-relevant relations and terms for similarity computation. Then the similarity between documents becomes a problem of formulating distance between two sub-graphs as shown in Fig. 4.1.

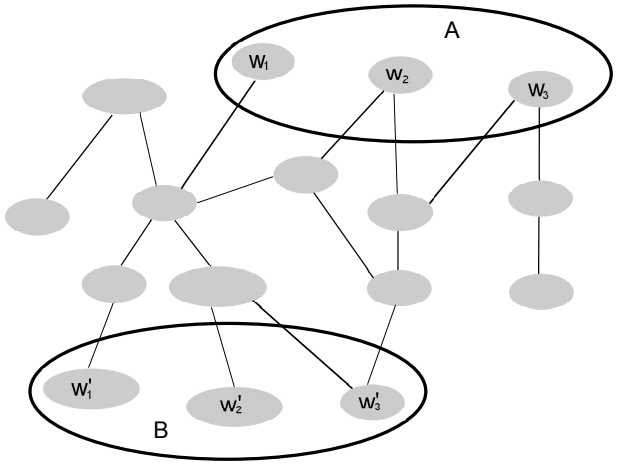


Figure 4.1: EMD for calculating distances/similarities between two documents A and B, which are represented as groups of nodes (words or phrases) in a network.

We will explore the use of the newly formulated measures to define similarity notions that can then be cascaded for use in a storytelling algorithm [82] or the stepping stones framework [115]. Storytelling/stepping stones are 'connect-the-dots' applications: the user supplies two documents and the algorithm is required to identify a series of intermediate documents that can bridge the seemingly disparate end points. We will investigate the use of distance measures such as Earth Mover's Distance [132] for this purpose. The weights of node and edges also need to be carefully formulated, and the frequency of terms and the type of grammatical relation will need to be taken into account. We will explore the use of the new similarity measures in both a cyber-intelligence dataset and a biomedical abstracts dataset.

4.2 Text Segmentation

Linear text segmentation is the task of segmenting text topically and capturing any inherent topic shift. Due to its importance and centrality in many natural language processing applications, including text summarization and information retrieval (e.g., linearly segmenting the snippets returned by search engine according to the topics), text segmentation has received significant attention in prior research.

Text segmentation algorithms can be classified according to their criteria for separating homogeneous segments from each other. Among state-of-the-art approaches, some of them use linguistic

details such as discourse cues [47], and others use statistical similarity measures [58, 26, 159]. Discourse cues are scarce in text but could be an important ancillary method. The statistical similarity measures assume that the topically homogeneous segments are lexically coherent, and use this property to detect topic shift. In the method of *TextTiling* [59], Hearst uses tf-idf, an information retrieval measure, to calculate the similarity between adjacent blocks, and then uses a sliding window approach to detect segment boundaries. Following this work, Choi [26] also uses a cosine similarity measure to calculate the between-sentence distance, but instead of the sliding window method, Choi conducts a divisive clustering of the sentence-similarity matrix. The *MinimumCut* segmentation algorithm [96], based on the image segmentation research of Shi and Malik [144], uses a graph-theoretic framework. A graph is constructed by modeling the sentences as the vertices, and the cosine similarity values as the edge costs, so text segmentation is viewed as a problem of partitioning a weighted undirected graph that minimizes the normalized-cut criterion. [71, 77] use dynamic programming techniques to identify the segmentation so that the cost is globally minimized.

The most obvious shortcoming of lexical iteration methods arises from their inability to capture the synonymy, antonymy, hyponymy, or polysemy of the language. Some algorithms overcome this problem with the help of semantic networks constructed from dictionaries [81], while some others [27, 104] use latent semantic analysis (LSA) to improve the computation of similarities. Ferret [39] exploits topic information from the text to strengthen the segmentation algorithm, and the algorithm assumes that the most representative words of each topic of a text occur in similar contexts, and that such topic words can be extracted by (unsupervised) clustering performed on the similarity matrix. Ferret uses these topic words to enhance the word reiteration-based segmentation, and this approach has been evaluated on artificial texts mixed with two topics, unlike the traditional testing dataset from Choi [26], which has more topics mixed.

Our work falls in the same vein as that of Ferret's in that we believe word iteration to be an insufficient basis for a segmentation algorithm. However, we do not exploit explicit topic information in advance, but rather use the entire text to construct a network that captures the similarities or differences between any two sentences. In more detail, as we can see from Figure 4.2, (1) firstly, each sentence of the input text is converted to a small dependency graph, and later merged together to form a bigger text level grammatical graph. (2) Secondly, we use the EMD algorithm to calculate the sub-graph similarity between any two sentences since any sentence can be represented as a sub-graph of the text level grammatical graph. Later, this similarity matrix is processed by using an edge-preserving smoothing scheme. (We adopt the bilateral filtering approach.). (3) At last, a new sliding window strategy is utilized to detect the segment boundaries. We discuss these three steps above in the following sections 2, 3 and 4, while in section 5 we talk about the evaluation.

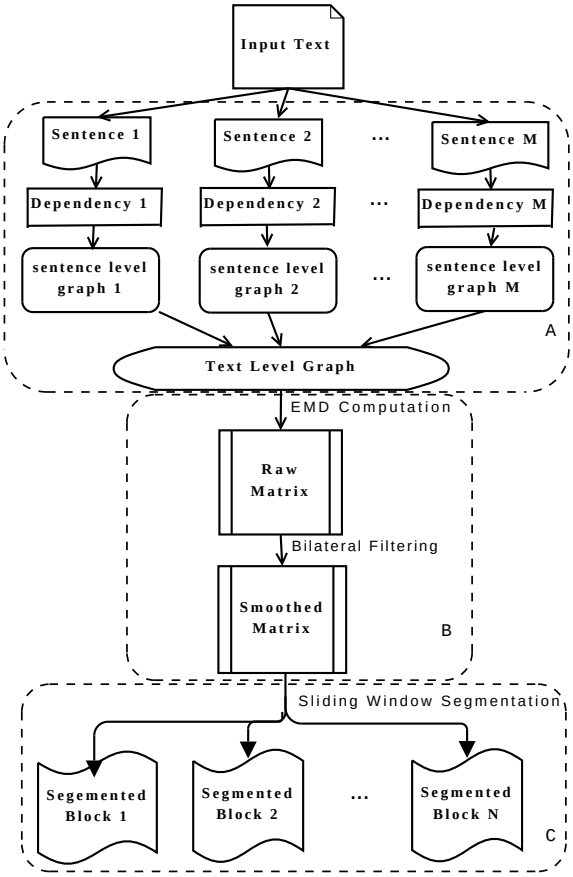


Figure 4.2: Flow chart for our segmentation system.

4.3 Building the Graph

Previous linguistic research has shown that the termination and alternation of lexical cohesion can be a strong indicator of topic shifting. For instance, *LCseg* [47] uses *lexical chains* [112] to represent the underlying text. The researchers then used a variant of an information retrieval method to compute lexical cohesion scores based on the lexical chain. Although our work is also based on principles of lexical cohesion, we use more complex graph structures than chains to capture deeper hidden relations between words and terms. Our approach to constructing the text level grammatical graph can be divided into three steps: we first break down sentences to small dependency parses; then we merge all these small sentence level dependency parses together to form a bigger graph; finally, we calculate the vertex weight and edge distance for the bigger text level grammatical graph.

In the first step, we build a grammatical relation graph from a single sentence, and in such a (small) graph, *context relationships* between the words, phrases, and concepts in the sentence are

Table 4.1: Rules for transferring the dependency

	grammatical relations or POS
deleted edges	aux, auxpass, punct, tmod, predet, num, quantmod, neg, det, cop, prt, possessive, expl, conj_and, conj_or.
retained vertices	FW, JJ, JJR, JJS, NN, NNP, NNS, RB, RBR, RBS, VB, VBD, VBG, VBN, VBP, VBZ, PRP, CD.

carefully described. While this small graph can be formed in several ways, we adopt the Stanford typed dependency parse [102] for the convenience of implementation. In our implementation, we use the collapsed typed dependencies (or grammatical relations) to construct the sentence level grammatical graph. Figure 4.3(a) shows the typed dependency parse output for a sample sentence.

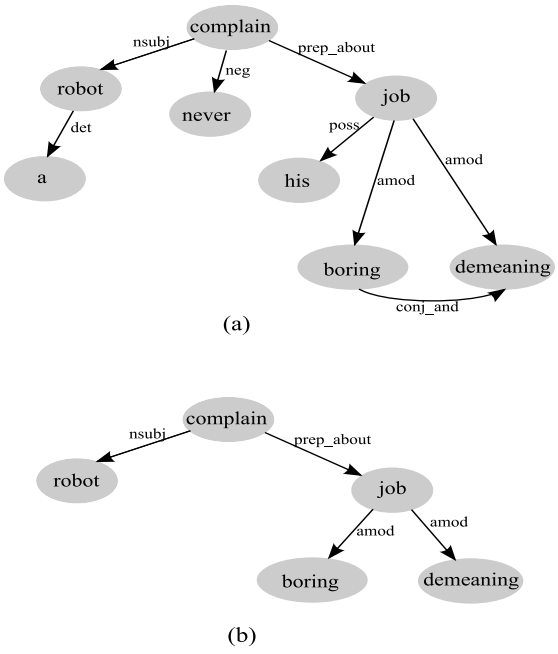


Figure 4.3: (a) A collapsed typed dependency parse for the sentence “A robot never complains about his boring and demeaning job". (b) The sentence level graph transferred from the parse in (a).

However, we do not retain the entire output from the parser as our sentence level graph. As we can see from Table 4.1, we apply some rules on the raw dependencies. First, some trivial edges (grammatical relations) are deleted, and all the non-head vertices attached to these edges will also be removed. Second, vertices with insignificant POS tags are removed as well, and when we remove these vertices, all their neighboring vertices will be connected automatically. In Figure 4.3(b), we give an example of the transferred and pruned sentence level graph.

In the second step, following common preprocessing standards, the tokenized words are firstly stemmed, and then we use a traditional stopword list to prune some vertices, and later we merge together all the small sentence level graphs generated from step 1 by connecting the common vertices in these graphs, leading to one or several bigger connected components known as the *text level grammatical graph(s)*. (The more related the topics are, the higher the possibility of obtaining a single connected text level context graph. Usually only one final text level graph will be generated.) A big text level graph consists of a number of vertices and edges, and reflects hidden relations identified from the whole text, meanwhile, sentences are represented as the subgraphs within the text level graph. Each vertex represents a linguistic concept (term, or phrase, etc.), and the edges between two vertices denote the affinity between two concepts. What's more worth mentioning is that by using the text level graph(s), we can easily plant any multi-word, or phrase level vertices besides the single word ones, so we can describe hierarchical relationships in our graphs. This is quite useful, and to some extent it solves the problem of comparing hierarchical concepts directly as long as we set the vertex weight in an appropriate way in the next step. The idea underlying these hierarchical relationships is straightforward: multi-word vertex matching usually plays a more important role in the similarity measure computation between two sentences[1].

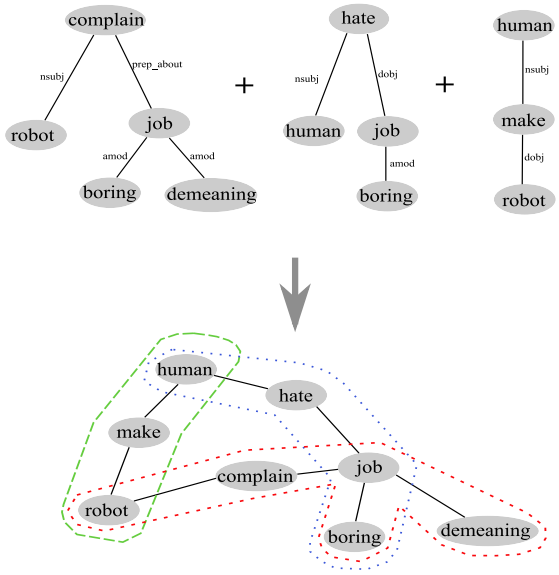


Figure 4.4: Three sentence level graphs are merged to form a singe one.

In the last step of constructing the text level graph, we add the weight values to the vertices and edges, serving as the signatures and ground distances for the EMD computation, respectively.

Vertex Weight: We apply a variant of the tf·idf [134] metric which is widely used in information retrieval area. Since the text may contain more than one topic, and these sentences are topically mixed together, we propose the **Segmented IDF** (SIDF) for the vertex weighting. The traditional

[1]We simply use the grammatical relation of "*NN*" to find the multi-word term or phrase.

IDF for term T_i in text segmentation can be calculated by $log(N/df_i)$, and N is the number of sentences containing T_i in the entire text, however, we do not use this number N directly. We group together all the neighboring sentences which contain concept T_i, so that the entire text is linearly divided into several groups, satisfying that one group has concept T_i, then its next neighbor does not have T_i ("blank group") and vice versa[2]. We calculate the IDF within these non-blank groups separately, and finally, sum up these group level IDF values in proportion to their corresponding groups' lengths:

$$W_j(T_i) = \log(L_j/df_j) \cdot (\log df_j + 1)$$

$$Weight(T_i) = \sum_{j=0}^{n} (W_j(T_i) \cdot \frac{L_j}{\sum_{j=0}^{n} L_j})$$

$$SIDF(T_i) = Weight(T_i) \cdot (\log Length(T_i) + 1))$$

L_j is the number of sentences in the jth non-blank group, df_j is the number of sentences containing T_i in jth group, $W_j(T_i)$ is the IDF value for term T_i in the jth group, and $Weight(T_i)$ is the normalized IDF value for term T_i in the entire text. Observe that we also consider the word number $Length(T_i)$ of concept T_i, and assign a higher value to the "longer" vertices (phrase, multi-word concept, etc.).

Edge Weight: We use the edge weight to simulate the distance from one vertex (concept) to another, and this also serves as the ground distance in the EMD computation. The distance is decided by both the vertices' degrees and the hop count between them. For a single edge (hop), the weight is calculated by:

$$Weight(e) = Degree(V_l) + Degree(V_r)$$

4.4 Sentence Similarity Computation

Our work is inspired by the earth mover's distance (EMD), which is based on a well-known transportation problem, and primarily used to calculate the similarity between two distributions in certain feature spaces, usually in the computer vision research area, and it is typically combined with image representation capabilities to design image comparison applications [133]. Recently, [165] use EMD as a measure for document similarity computation, which in spirit is similar to our sentence similarity computation. [174] also proposed EMD-based similarity search in multimedia databases.

We hasten to add that our work, however, is not a straightforward application of the EMD measurement. Since we represent the entire text as a graph, the key issue here is to construct the graph in such a manner that it can provide the *ground distances* and the *signatures* for the EMD computation. Our experiments show that the graph-building scheme we propose is suitable for the EMD computation, and the resulting sentence similarity measure performs quite competitively.

[2]In our implementation, we do not strictly require that the neighboring two sentences must both have the concept term in the non-blank group, however, a maximum gap of 3 is allowed.

4.4.1 Raw Similarity Matrix Computation

In our algorithm, we use the EMD similarity measure to calculate the distance between two sentences. As we already mentioned, EMD is based on the solution to the transportation problem of finding a least-expensive flow of goods from the suppliers to the consumers which can satisfy the consumers' demand. Here we model the graph distance measure problem as a transportation problem. When we compare two sentences, we can think of "moving" the terms' meaning in one sentence towards the other, and if two sentences have a closer meaning to each other, the distance between them should be smaller, and hence the transportation cost should also be lower.

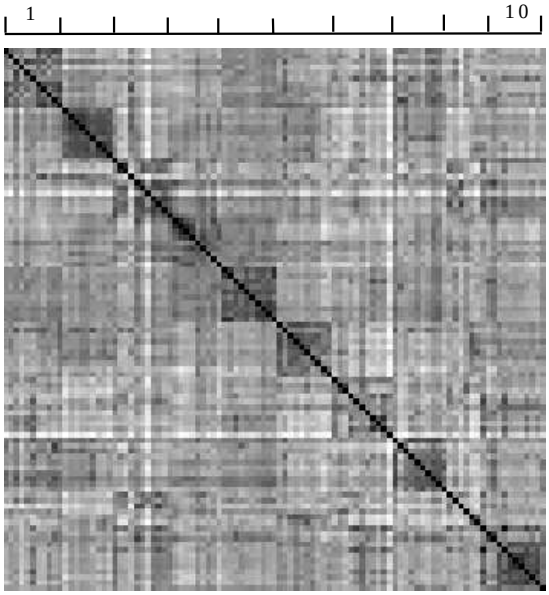


Figure 4.5: Symmetric sentence similarity matrix plot for a sample text from Choi's synthetic testing dataset. This dataset has 10 synthetic segments. In this matrix, each point represents the similarity between the row sentence and the column sentence. The darker the color is, the higher similarity (smaller distance) they have. The black blocks along the diagonal illustrate the segments because usually sentences in the same segment have similar meanings and high similarity.

During the computation of EMD, we use two signatures

$$A = (p_1, w_{p_1}), \ldots, (p_m, w_{p_m})$$

and

$$B = (q_1, w_{q_1}), \ldots, (q_n, w_{q_n})$$

to represent a sentence pair, where m, n are the numbers of terms/concepts in these sentences, w_{p_m}, w_{q_n} are the weight values of the vertices in these two sentences (total vertex weights of each sentence are normalized to 1 to guarantee that EMD values are comparable), and we use distance

matrix $D = [d_{ij}]$ to store all the ground distances (transportation cost) of vertex pairs. We find a flow $F = [f_{ij}]$ where f_{ij} denotes the flow assigned to the path from vertex i to vertex j, and such flow satisfies the minimal total cost requirement on $\sum_{i=0}^{m}\sum_{j=0}^{n} f_{ij}d_{ij}$. Finally, when the two sentences A and B are in the same text level graph G, the earth mover's distance between these two sentences is normalized by the total flow:

$$EMD(A,B) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} f_{ij}d_{ij}}{\sum_{i=0}^{m}\sum_{j=0}^{n} f_{ij}}$$

$$Sim(A,B) = \begin{cases} EMD(A,B), & \text{A,B are both in G;} \\ -1, & \text{otherwise.} \end{cases}$$

At the same time, since we have the perspective of hierarchical relationships in the graph, when the two sentences have multi-word term/concept matching, we simply discard the single-word vertices in the signatures which may duplicate the multi-word version and lead to a duplication. Figure 6.2 shows the similarity matrix generated from one of Choi's artificial testing texts.

4.4.2 Matrix Smoothing

As we described in Figure 4.2, we smooth the raw similarity matrix before the segmentation, and in order to smooth the data inside segments while still keeping the edges, we apply bilateral filtering on the raw similarity matrix. Bilateral filtering is ubiquitously used in computational photography and computer graphics areas. It was proposed by [157], and the main purpose is to average the regions while preventing averaging across the edges. Figure 4.6 shows different smoothing scales on a raw similarity matrix.

4.5 Linear Segmentation via Sliding Windows

There remains the problem of linearly segmenting the text given the similarity matrix. Currently many methods are used, such as dynamic programming and minimum cut. However, we use the sliding windows to detect the topic shift as the most direct way. Like in *TextTiling*, F06T, and *LCseg*, our segmenter also uses fixed size sliding windows. *TextTiling* and *LCseg* use the cosine measure, and F06T uses the Dice coefficient, to compute the similarity values between the left window and the right window near each sentence break, and when the sentence break is also the segment break, this value usually will be the lowest. However, those methods assume that all segments have a similar cohesion level. In a real text, some segments may not have a clear topic, and hence such segments may only have a low level of cohesion. See Figure 6.2. the 8th and 10th segments can be clearly identified, but the 9th can not. To address this problem, we propose a variant of sliding window strategy, and also a new concept of **Average Similarity Difference (ASD)** for measuring the scale of topic shift. The idea underlying is relatively straightforward,

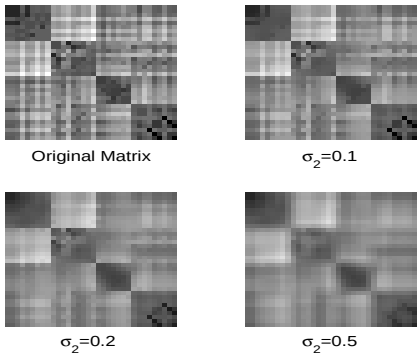


Figure 4.6: Bilateral filtering effects on raw similarity matrix, with different parameters. Our experiments shows that the raw similarity matrix is not sensitive to the parameters of filter window size W and spatial-domain standard deviation σ_1, however, the intensity-domain standard deviation σ_2 significantly affects the smoothness. We adopt $\sigma_2 = 0.1$ in our experiment.

since we already smooth the similarity matrix. As long as we detect any sharp change, we can mark it as a segment break. Figure 4.7 shows the sliding windows near the segment break after the *kth* sentence.

As in Figure 4.7, we use the distance values inside the sliding windows to compute the ASD. Each sentence break has both a left window W_L and a right window W_R (we set the window size as 6), and each window has two rows: W_L has rows R_A and R_B, while W_R has rows R_C and R_D. We sum all the distance values inside each row, and compute the ASD value for each window as:

$$ASD(W_L) = \frac{\sum R_B - \sum R_A}{\sum R_A}$$

$$ASD(W_R) = \frac{\sum R_C - \sum R_D}{\sum R_D}$$

The ASD value at sentence break k is:

$$ASD(k) = Max(ASD(W_L), ASD(W_R))$$

The main difference between our algorithm and other sliding window algorithms is: (1) We do not compare two windows of sentences to get the value; (2) We only use the bigger ASD value from one window, either the left, or the right. The reason is that the segments in the text may have different lexical cohesion levels. Imagine that in Figure 4.7, if the bottom right segment has lower cohesion level, these data grids inside this segment will be in a light color. Also, the difference between data in row A and row B, can be much bigger than the difference between data in row C and row D. However, sharp topic shift (similarity difference) in either window is enough for marking a segment break (boundary).

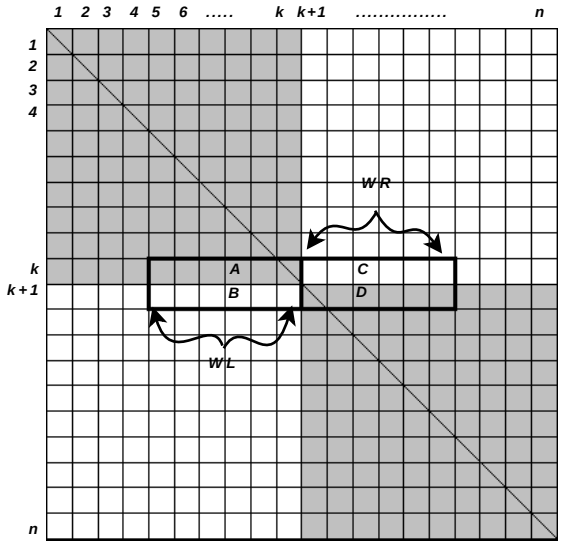


Figure 4.7: Sliding windows used to compute the ASD value at sentence break. Data grid with light color indicates long distance between sentences (larger value).

We only consider ASD values above zero. A bigger ASD value means a bigger topic shift. We identify the boundaries by applying a threshold to the ASD values. As we can see from Figure 4.8, we compute the threshold in a similar way to *LCseg* and F06T, by using the mean (μ) and standard deviation (σ) of the ASD values. We pick the sentence breaks with ASD values above the threshold $\mu + \alpha \cdot \sigma$ (we set $\alpha = 0.6$ in our experiment) as the potential boundaries. Finally, we use these boundaries to segment the text, and ignore the segments which are too small.

4.6 Evaluation

4.6.1 Error Metric

As in most of the text segmentation algorithms, precision or recall is not used for evaluation, instead, we evaluate our algorithm by using the error metric P_k[3] proposed in [7], which is designed to overcome the disadvantages of precision or recall. Later, another variant of the error metric P_k called *WindowDiff* (WD) [125] was proposed, that is intended to address the insufficiencies of the

[3] P_k measures how well the identified boundaries match the real reference boundaries by checking the false negative mistake (when a true break is missed by the algorithm) and false positive mistake (when a hypothesized break is not a real break) of an algorithm. If **ref** is the correct segmentation and **hyp** is the result of a segmentation algorithm, "the number P_k(**ref**, **hyp**) is the probability that a randomly chosen pair of words a distance of k words apart is inconsistently classified; that is, for one of the segmentations the pair lies in the same segment, while for the other the pair spans a segment boundary."[7] So smaller P_k means better performance.

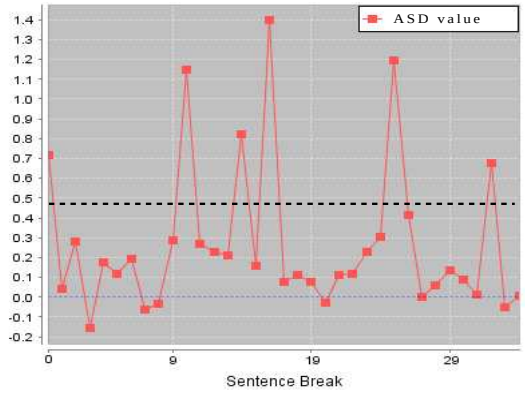


Figure 4.8: Our segmentation algorithm applied on a synthetically generated text piece in our dataset, which has 4 segments. The x-axis represents sentence break indices, the y-axis represents the ASD values at the breaks, and the black horizontal dotted line shows the threshold. In this example, the error metric of our algorithm is P_k=0, while P_k in U00 is 0.14 and in C99 is 0.15.

P_k metric. We use both of these two metrics in our evaluation experiments.

4.6.2 20 Newsgroups Dataset

Most of the previous segmentation algorithms use the standard evaluation dataset proposed by Choi [26], which is synthesized from the Brown corpus, however, Choi's texts are artificial and for each text there are too many different topics. Ferret [39] proposed another way to create the evaluation texts to address this problem: Ferret adapted Choi's framework for having more realistic texts by changing the way to select the source documents. Ferret did not choose the segments from different documents for synthesizing text, but only from two source documents, and each document is split into several segments of a certain size. Ferret then alternatively concatenated segments from these two documents to form a new text. Inasmuch as we use both lexical cohesion and the topic information, which is similar to Ferret's method, we evaluate our algorithm through corpora created in the same way as, but much bigger than F06T's evaluation corpora. We created our evaluation corpora based on the public *20 Newsgroups* [86] dataset[4], consisting of about 20,000 newsgroup documents, and partitioned across 20 different newsgroups in different topics. Some of these newsgroups are very similar, while others are highly unrelated. We choose four groups among them ("*rec.autos*", "*rec.sport.baseball*", "*rec.sport.hockey*" and "*talk.politics.guns*"), two of these four groups are more related ("baseball" and "hockey") and belong to the same category of sport, whereas others are in different topics according to the subject matter. We cleaned these newsgroup documents by removing duplicated and crossposted documents, and we also removed the headers of "*From*" and "*Subject*" for each document. Table 4.2 shows the main characteristics

[4]http://kdd.ics.uci.edu/databases/20newsgroups

of newsgroup documents in the four categories.

Table 4.2: Information about our evaluation corpora

	auto	**baseball**	**hockey**	**gun**
src doc	990	994	999	910
valid doc	347	275	293	304
tokens/doc	153	160	153	183
sentences/doc	13	14	13	15

As in F06T, each document in the newsgroup was split into several segments (from 6 to 11 sentences per segment), and an evaluation text was built by alternatively concatenating segments from two different newsgroups, so we obtained $\binom{4}{2}$ combinations and hence 6 new categories in our evaluation dataset (auto_baseball (AB), auto_guns (AG), auto_hockey (AH), baseball_hockey (BH), guns_baseball (GB), guns_hockey (GH)). To some extent, the text we create can be regarded as a mixed multi-thread online discussion record. Moreover, texts in our dataset have a range of different sizes (4 segments/text, 6 segments/text, 10 segments/text, and 20 segments/text), which has a wider coverage[5].

We compared our algorithm (EMD) with several other state-of-the-art algorithms[6]: U00 [159] uses statistical methods for segmentation, C99 [26] is the one that accompanies Choi's testing dataset, MinCut [96] formalizes segmentation as a graph-partitioning task,[7] and F06T [39] is Ferret's algorithm using the topic information. The number of segments is unknown to these algorithms in our experiments except MinCut. As the results show in Tables 4.3, 4.4, 4.5 and Figure 4.9: (1) Our algorithm is significantly better than MinCut and F06T for all corpora. (2) For the first group of testing corpora (see Table 4.3), in which each text has a small size of segments (2, 3 and 4 segments/text), our algorithm is significantly better than all other algorithms. (3) For the second group of testing corpora (see Table 4.4), our algorithm can still beat most traditional methods which primarily are based on word iteration.

The notable point is that our algorithm performs very well especially for lean text which is very short and has only very few segments. In this situation, the other traditional algorithms, based mainly on word iteration, without linguistic analysis, have difficulty capturing the hidden relationships between words, and hence the statistical information may not be enough to identify the topic boundaries. Experimental results on the first group of lean text corpora proves our graph-based algorithm starts to function even when the text size is very small. Moreover, our algorithm and C99 both are more stable than U00 and are less sensitive to segment number changes than the others. As in Figure 4.10, U00's performance significantly varies when the segment number of the text changes, although it works very well when the number is near 10.

[5] All texts in Choi's and Ferret's test corpora have exactly 10 segments. We are concerned that algorithms tested on corpora with a predefined number of segments may overfit.

[6] Our dataset and implementation can be dowloaded from: http://sites.google.com/site/textseg/

[7] http://people.csail.mit.edu/igorm/acl06code.html

Table 4.3: Detailed evaluation result (P_k) of segmentation algorithms on the newsgroup corpora with small segment number (2, 3 and 4). $p-value$ is associated with *T-test* between EMD and C99.

P_k (%) 2 segments/text						
	AB	AG	AH	BH	GB	GH
C99	29.39	31.07	26.61	29.44	30.64	25.20
U00	54.63	56.15	54.65	54.34	55.59	55.86
MC	31.34	29.51	29.17	32.15	27.79	27.74
F06T	36.56	39.35	34.75	38.08	37.53	37.00
EMD	**22.80**	**23.55**	**20.26**	**27.27**	**22.12**	**21.09**
p-val.	0.001	1e-4	4e-4	0.17	1e-5	0.03
P_k (%) 3 segments/text						
	AB	AG	AH	BH	GB	GH
C99	27.31	28.40	25.40	28.35	24.42	22.56
U00	37.92	37.90	36.41	38.95	37.77	38.00
MC	35.84	34.03	36.81	38.25	35.94	34.50
F06T	34.37	36.72	32.35	34.94	34.32	32.75
EMD	**21.61**	**23.97**	**24.04**	**26.00**	**21.13**	**20.35**
p-val.	2e-4	8e-4	0.17	0.07	0.01	0.08
P_k (%) 4 segments/text						
	AB	AG	AH	BH	GB	GH
C99	24.71	25.89	23.51	27.37	24.22	22.01
U00	27.65	29.64	25.49	29.01	27.22	26.19
MC	38.24	37.14	37.76	39.76	36.72	37.31
F06T	31.95	34.50	30.43	33.96	32.30	30.60
EMD	**22.44**	**23.33**	**21.79**	**24.53**	**21.53**	**18.43**
p-val.	0.018	0.009	0.05	0.008	0.014	.0005

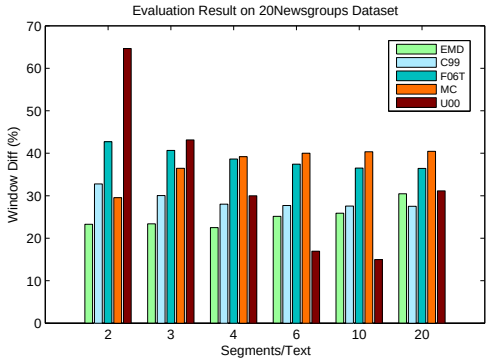


Figure 4.9: Comparison result of different algorithms on the 20 newsgroups dataset.

4.6.3 Google Blog Search Result Dataset

We also evaluate our algorithm on the Google blog search result. As in Figure 4.11, the blog search result snippets are typical lean text. (Another good example of lean text could be the news feed

Table 4.4: Detailed evaluation result (P_k) of segmentation algorithms on the newsgroup corpora with segment number above 6. $p-value$ is associated with *T-test* between EMD and C99.

P_k (%) 6 segments/text						
	AB	AG	AH	BH	GB	GH
C99	25.02	27.15	**22.33**	26.68	24.36	21.87
U00	16.36	18.51	14.20	17.16	16.65	15.53
MC	38.71	35.81	37.57	39.47	35.97	37.61
F06T	29.42	33.00	29.50	30.68	29.95	31.97
EMD	25.08	**25.24**	24.20	26.18	25.44	21.93
p-val.	0.48	0.042	0.048	0.33	0.17	0.47
P_k (%) 10 segments/text						
	AB	AG	AH	BH	GB	GH
C99	26.76	26.52	22.42	30.01	23.4	23.46
U00	15.44	15.65	12.85	19.09	13.38	12.94
MC	37.23	37.23	38.01	39.75	35.09	36.56
F06T	30.62	31.05	27.67	31.17	30.37	28.42
EMD	26.69	25.21	22.16	29.81	24.37	24.3
p-val.	0.479	0.186	0.424	0.437	0.17	0.22
P_k (%) 20 segments/text						
	AB	AG	AH	BH	GB	GH
C99	30.95	26.48	**24.13**	**26.51**	**23.07**	**23.86**
U00	31.69	32.25	31.72	32.00	29.23	29.83
MC	38.24	36.22	36.62	40.22	35.68	36.93
F06T	31.62	30.13	27.19	30.77	30.11	29.13
EMD	30.67	28.91	28.71	34.29	28.78	28.70
p-val.	0.49	0.105	0.016	7e-4	0.002	0.004

Table 4.5: Average WindowDiff result of algorithms on the 6 datasets with different segment number (per text). *p-value* is associated with *T-test* between EMD and C99.

WindowDiff (%)						
# seg	C99	U00	MC	F06T	**EMD**	p-val.
2	32.78	64.64	29.55	42.71	**23.28**	$\sim$0
3	30.05	43.13	36.45	40.67	**23.40**	$\sim$0
4	28.03	29.99	39.20	38.65	**22.49**	$\sim$0
6	27.70	16.96	40.01	37.43	**25.15**	2e-7
10	27.58	14.98	40.36	36.51	**25.89**	.002
20	**27.52**	31.16	40.46	36.43	30.46	1e-4

text stream from social network websites, such as Facebook.) Compared to the traditional synthetic evaluation datasets which are constructed by mixing several pieces of text, the blog search result contains natural boundaries between different snippets of different topics for evaluation. Figure 4.12 shows that our algorithm has outstanding performance (the lowest Window Diff value) in detecting the topic boundaries.

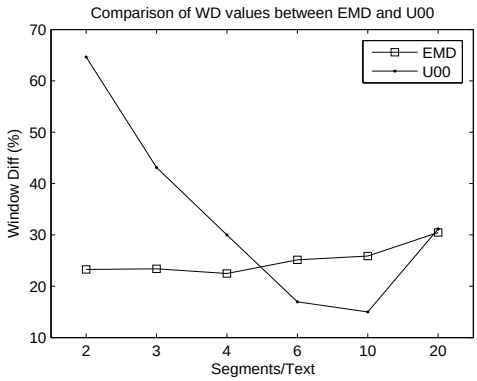


Figure 4.10: Comparison between EMD and U00.

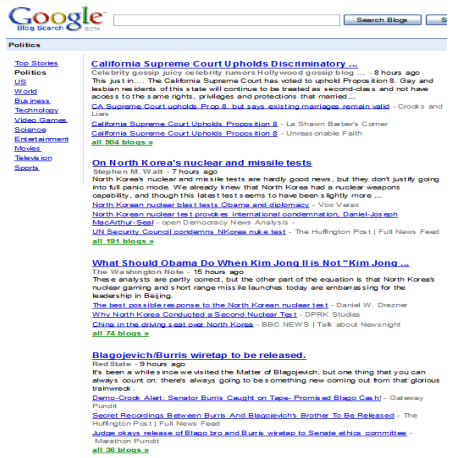


Figure 4.11: Google blog search homepage.

4.7 Conclusion and Future Work

In this chapter, we have proposed a new algorithm for text segmentation based merely on text grammatical graphs, in contrast to external knowledge such as ontology or dictionaries. We have introduced a new method using the earth mover's distance to measure the similarity between sentences. Moreover, we created new evaluation corpora from the real internet newsgroup text, and we also use real Google blog search result pages as an evaluation dataset. Our experimentation demonstrates that EMD-based text segmentation yields competitive performance.

Our current implementation pipeline can be improved in many ways. First, pre-computed graphs from larger scale corpora can more accurately reflect the real relationships between words and terms, and hence improve the accuracy of the sentence similarity measurement. Second, some simplified approach other than using deep grammatical relationships might suffice, such as the

Figure 4.12: Comparison result of different algorithms on the blog search result dataset.

lexical chain. Meanwhile, anaphora resolution analysis and external domain knowledge such as cue phrases or lexical databases like WordNet[8] can be easily plugged into our framework.

[8]http://wordnet.princeton.edu

Chapter 5

Dependency Parses for Query Expansion: Applications to Drug Interaction Search

Given a drug name under development, how can we find all the other drugs or biochemical compounds that it might interact with? Early answers to this question, by mining the literature, are valuable for pharmaceutical companies, both monetarily and in avoiding public relations nightmares. Inferring drug-drug interactions is also important in designing combination therapies for complex diseases including cancers. We study this problem as one of mining linguistic cues for query expansion. By using (only) positive instances of drug interactions, we show how we can extract linguistic cues which can then be used to expand and reformulate queries to improve the effectiveness of drug interaction search. Our approach integrates many learning paradigms: partially supervised classification, association measures for collocation mining, and feature selection in supervised learning. We demonstrate compelling results on using positive examples from the DrugBank database to seed MEDLINE searches for drug interactions. In particular, we show that purely data-driven linguistic cues can be effectively mined and applied to realize a successful domain-specific query expansion framework.

5.1 Background

Given a drug under development, what are other drugs or biochemical compounds that it might interact with? History abounds with instances where this question had been investigated incompletely, causing severe personal, monetary, and professional losses. Two recent examples serve to illustrate this aspect well. In Sep. 2004, drug maker *Merck* voluntarily recalled their anti-inflammatory drug *Vioxx* because they discovered that patients taking Vioxx faced heightened risk of heart attacks. In May 2009, it was discovered that older men using the *Flomax* drug (to address their urinary tract problems) are twice as likely to experience 'floppy iris syndrome,' a condition that can cause inflammation around the eye, retinal detachment, and other serious side effects [9].

Thus there is great interest in mining for drug interactions before product development, clinical trials, and market releases.

Formally, a drug interaction can be defined as "the pharmacological or clinical response to the administration or co-exposure of a drug with another substance that modifies the patient's response to the drug" [141]. While the intent of inferring drug interactions is often to avoid them, sometimes it is actually desirable to encourage such interactions. For instance, in treating complex diseases such as cancers [127], multiple, or multi-component, drugs are usually administered in order to enhance combinatorial selectivity [42]. Here the different components work cooperatively to inhibit (or activate) a protein or protein network of interest in a diseased tissue, but are not particularly toxic for normal tissues.

Many computer-aided systems for exploring drug interactions have been established for clinical decision making. One such system [80] uses the CYP3A cytochrome (a key component of many drug metabolism pathways) as the focal point to rank potential drug interactions. Another interesting system exploits the inherent network structure of drug interactions [151]. The key issue in these systems (and all others) is the *completeness* of their drug interaction information and ensuring that they stay current with published literature. On one hand, it is inconvenient for a clinician to search the scattered literature for information about a specific drug. On the other, database maintainers are concerned with issues of soundness, coverage, and trustworthiness.

The aim of our work is, hence, to design a framework for automatically identifying drug interaction sentences from large online corpora. Given a specific drug name such as *darbepoetin alfa*, a naive search for *darbepoetin alfa* in public corpora often gives a poor starting point, yielding hits such as the usage of *darbepoetin alfa* in the treatment of *anaemia*. By using state-of-the-art query expansion algorithms [110, 117, 175, 176], *darbepoetin alfa* gets expanded to *darbepoetin alfa anemia* or *darbepoetin alfa cancer*, which still leads to results of usage and is unsuitable for drug interaction identification. Even a manually formulated query, like *darbepoetin alfa interaction* or *darbepoetin alfa use together*, gives results such as "Recombinant human epoetin beta in the treatment of renal anemia"[1] – they still do not cover drug interactions very well. In this work, we turn to designing a domain-specific query expansion capability (similar in spirit to [43, 94, 118]) by mining linguistic cues to combat this lack of specificity.

There are several research issues to be considered. Admittedly, one way to expand queries is to 'learn to expand,' i.e., to mine linguistic cues from a training dataset and use them to expand future queries. However to conduct such learning, negative examples are required and these are quite rare to come by. So the first research issue is to be able to work with only positive examples. Second, the precise nature of linguistic cues can involve just a single term (e.g., *darbepoetin alfa* can be expanded to *darbepoetin alfa coadministration* or a complete collocation (e.g., *darbepoetin alfa* might get expanded to *darbepoetin alfa concomitant use*). Mining cues and collocations is the second research issue and association measures have to take into account the complexity of syntactic constructs in the published literature. Finally, in order for the mined cues and collocations to be successfully used, they will have to be ranked so that they prominently accompany positive

[1] http://www.ncbi.nlm.nih.gov/pubmed/18488073

examples but not negative examples. Feature selection to help separate out the cues is hence important.

Our primary contributions are:

1. We present a domain-specific solution to drug interaction search that fuses multiple learning paradigms: partially supervised classification, association measures for collocation mining, and feature selection in supervised learning. These paradigms are used, respectively, to overcome lack of negative examples, to identify promising cues and associations between them, and to rank cues so that they lead to confirming instances of drug interactions. We highlight here that our approach works without requiring any external thesaurus or ontology. Nevertheless, we refer to it as a domain-specific solution due to its reliance on corpora specific to the domain and because we haven't evaluated its utility in other domains.

2. We propose a new collocation mining approach which utilizes both positive and negative datasets. Hence the cues we mine are not just co-located but also possess good discriminative properties.

3. We apply our framework on the DrugBank [2] database (which features only positive instances of drug interactions) and show how it can be used to seed MEDLINE [3] searches for drug interactions. To the best of our knowledge, this is the first integrated framework for query expansion in drug interaction search.

5.2 Framework Overview

Fig. 5.1 gives an overview of our framework which is based on text mining and data-driven collocation techniques. As stated earlier, we use linguistic cues for query expansion and aim to identify new drug interaction sentences which can then seed DrugBank evolution. We begin by using existing drug interaction description sentences in DrugBank as a positive dataset, and search MEDLINE with only drug names as queries. We then aim to construct a negative dataset as a subset of the search results. Since a manually labeled negative dataset is expensive, we construct the negative dataset by using the partially *supervised classification* method, specifically the 'learning from positive and unlabeled examples' paradigm (a.k.a. the LPU method)[4] [92]. LPU is a classification system based on positive and unlabeled datasets, and it learns the negative examples automatically. These positive and negative datasets serve as starting points for two types of analysis, as shown in Fig. 5.1. First, based on these training datasets, we design a novel algorithm to extract cue words that serve as confirming evidence for drug interaction relations. We use text feature selection approaches to extract single-term cues. We also extract multi-word cues by using and

[2] http://www.drugbank.ca

[3] http://www.nlm.nih.gov/

[4] http://www.cs.uic.edu/∼liub/LPU/LPU-download.html

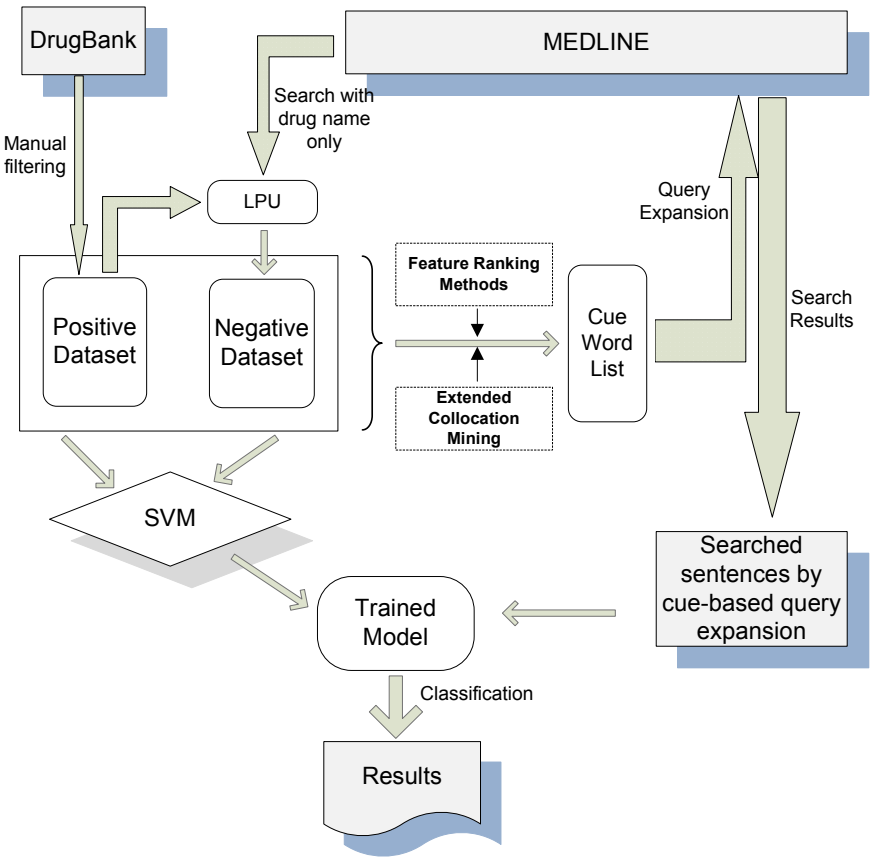


Figure 5.1: Flowchart for our drug interaction identification system.

assessing 13 different association measures for collocation mining. All these cue words are used to query-expand, to help search for new drug interactions in MEDLINE. Meanwhile, the new search results from the expanded queries are segmented into sentence level units and classified by a state-of-the-art SVM, which is trained on the initial data sets (this is their second use, referred to earlier). Experimental results show that our system works in an effective and efficient manner. We perform multi-faceted evaluations. For instance, we show that the negative training dataset automatically generated is comparable to a manually labeled one and any differences do not have any influence on the accuracy of the overall mining process. We also compare our findings with information about existing drug interactions in DrugBank.

5.3 Related Research

Query expansion has a rich history of background research. Traditional query expansion uses term relationships between the original term and the expanding term. Global analysis [4, 72,

Table 5.1: Comparison of our approach with other query expansion methods.

	Use local analysis	No user interaction	No external knowledge	Use linguistic property	Domain-specific
[110, 117, 175, 176]	✓				
[14, 30, 45, 168]	✓		✓		
[78, 162]		✓	✓	✓	
[94]	✓	✓			✓
[43, 118]	✓	✓	✓		✓
Our method	✓	✓	✓	✓	✓

130] employs global statistical information gathered based on co-occurrence information from the entire collection, whereas local analysis [110, 117, 175, 176] is conducted using approaches modeled after pseudo-relevance feedback, i.e., using the top ranked relevant documents. Some other approaches [14, 30, 45, 168] mine logs of past query usage to construct term relationships. Significant work also has gone into incorporating external knowledge [12, 63, 93, 97]. A review of ontology-based query expansion is given in [12]. Works such as [78, 97, 162] use collocation techniques from NLP which also are based on term-term co-occurrence information. In many of these works, frequency of co-occurrence is used as a surrogate for relevance and this leads to both false positives and false negatives [121]. This is especially true in drug interaction search. A detailed comparison with our approach and other query expansion methods is shown in Table 5.1. As is clear we aim for an automated, agnostic method that exploits local linguistic cues. The training imparted to our system from drug interaction sentences endows it with the domain-specific query expansion facility. Other domain-specific approaches such as [43, 94, 118] are either focused on different application needs and/or they do not exploit collocations as heavily as is done here.

5.4 Mining Linguistic Cues

The most important part of our system involves the extraction of linguistic cues which is covered in detail here.

5.4.1 Single-term Cue Word Extraction

We employ three criteria functions to rank all the tokens/terms appearing in the whole dataset. We classify sentences into two categories: positive and negative, according to whether they contain any drug interaction information. We consider the top 50 tokens ranked by each method as our single-term cues.

Mutual information (MI): The mutual information between term t_i and category *c* is given by:

$$MI\left(t_i, c\right) = \sum_{t_i \in \{0,1\}} \sum_{c \in \{+,-\}} P\left(t_i, c\right) \log \frac{P\left(t_i, c\right)}{P\left(t_i\right) P\left(c\right)}$$

Fisher kernel: We use the variant of the Fisher kernel as defined in [24], where the *F-score* for the *i*-th term is defined as:

$$F(i) = \frac{\left(\overline{x}_i^{(+)} - \overline{x}_i\right)^2 + (\overline{x}_i^{(-)} - \overline{x}_i)^2}{\frac{1}{n_+ - 1}\sum_{j=1}^{n_+}\left(x_{j,i}^{(+)} - \overline{x}_i^{(+)}\right)^2 + \frac{1}{n_- - 1}\sum_{j=1}^{n_-}\left(x_{j,i}^{(-)} - \overline{x}_i^{(-)}\right)^2}$$

Here, n_+ and n_- are the number of positive and negative sentences in the training dataset, $x_{j,i}^{(+)}$ and $x_{j,i}^{(-)}$ represent the *i*-th feature value in the *j*-th sentence in the positive and negative datasets, respectively. $\overline{x}_i$, $\overline{x}_i^{(+)}$, $\overline{x}_i^{(-)}$ are the average values of the *i*-th term's weight computed across the whole dataset, the positive dataset, and the negative dataset, respectively. The term weighting scheme thus influences the Fisher kernel; in our experiments, we use a simple binary weighting scheme so that the weight for a term is 1 if it appears in the sentence, 0 otherwise.

Relative frequency: The relative frequency *rf* for the *i*-th term was proposed in [84] as a ranking function:

$$\text{rf}_i = \log(2 + \frac{a_i}{c_i})$$

where a_i is the number of positive sentences in which this term appears, and c_i is the number of negative sentences it appears in.

5.4.2 Bigram Cue Word Generation

Multi-word cues, such as bigrams, require more sophisticated means than presented above. We extend *collocation* mining approaches from NLP to extract bigram cue words, ensuring properties of both **coexistence** and **discriminativeness**. Coexistence requires that the two terms in a bigram cue co-occur with each other a lot, while discriminativeness means these two terms in the bigram can be used to identify the target sentences.

Collocation

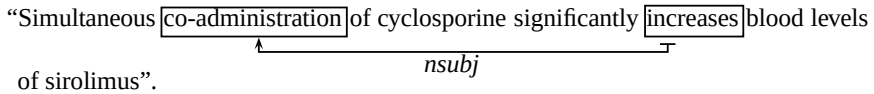
Collocation, as a linguistic concept, was introduced by J. R. Firth [40]. There are multiple definitions for what a collocation is, drawing upon different perspectives and the needs of specific applications. Smadja views collocations as lexical clusters that are domain-specific, context-recurrent, and cohesive [145]. Manning and Schutze [99] mention three attributes of typical collocations: *non-compositionality*, *non-substitutability*, and *non-modifiability*. Wermter and Hahn [173] provide a useful grouping of collocations into three classes:

- *Idiomatic Phrases*. In this class, the meaning of the collocation cannot be determined by the literal definition of the phrase components themselves, but refers to a metaphorical or figurative one. For example, the figurative meaning of "Wall Street" is the American financial market, and is unconnected to either 'Wall' or 'Street.'

- *Support Verb Constructions/Narrow Collocations*. In this class, at least one component contributes to the overall meaning of the collocation expression. For example, “Sunday driver” means one who drives slowly.
- *Fixed Phrases*. This class denotes expressions whose components are all involved in the contribution of the overall meaning (e.g., the collocation “Christmas Eve”).

A collocation useful for drug interaction search is “*concomitant administration*”, where the occurrence of the collocation with a drug name is likely evidence of a drug interaction sentence.

The typical way to study collocations within a text dataset is to measure the co-occurrence frequency using statistical association measures [37] and use this information to identify three categories: *surface co-occurrence*, *textual co-occurrence*, and *syntactic co-occurrence*. Surface co-occurrence involves two words that appear within a certain distance, textual co-occurrence requires that the two words appear within the same textual unit, and syntactic co-occurrence requires the existence of syntactic relations between the terms. In our study, we use the restrictive notion of syntactic co-occurrence since it avoids setting a arbitrary distance for surface co-occurrence, and also avoids indirect and accidental co-occurrences. Further, compared to superficial co-occurrences, terms that are physically close to each other could be linguistically far apart, and hence should not be considered together as co-occurrence, whereas terms that are linguistically close could be physically remote. In the sentence ‘Simultaneous co-administration of cyclosporine significantly increases blood levels of sirolimus’, the word “co-administration” is not close to the word “increases”, but serves as a nominal subject, and this combination is quite frequent in drug interaction sentences (see below).



Co-occurrence database

To obtain syntactically bound co-occurrence information, we utilize dependency parsing, specifically using the Stanford typed dependency parser [102]. We use collapsed typed dependencies (a.k.a. grammatical relations) to represent specific co-occurrence patterns, such as *adjective + noun*, *adverb + verb*, etc. However, we do not retain all possible collapsed typed dependencies, but only those with grammatical relations like *obj*, *dobj*, *iobj*, *pobj*, *subj*, *nsubj*, *nsubjpass*, *csubj*, *csubjpass*, *amod*, *advmod*, and *nn*. A typical dependency parse of a sample sentence from our positive dataset is shown in Fig. 5.2.

We treat grammatical relations as 2-tuple co-occurrence pairs, and store them into a co-occurrence database as candidates for bigram cues. We build 2-tuple co-occurrence records for both positive and negative datasets. As an example, 2-tuples from a sample sentence are shown in Table 5.2. Moreover, all the words are stored in lemmatized format, since words in the sentences could be

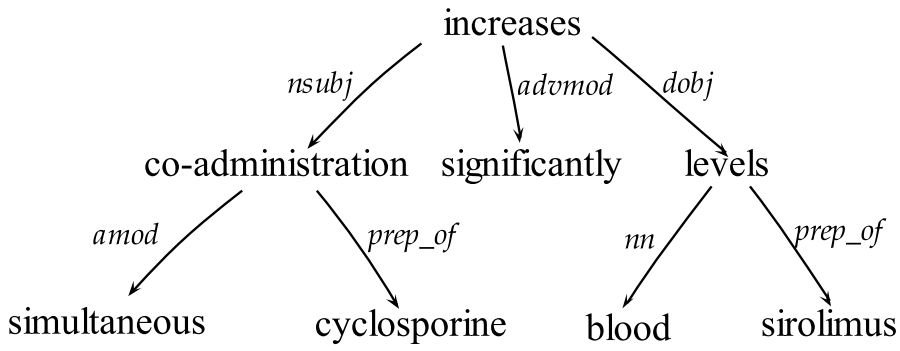


Figure 5.2: Dependency parse from the positive sentence “Simultaneous co-administration of cyclosporine significantly increases blood levels of sirolimus”.

in inflected forms. Lemmatization is used (e.g., *synchronized* is lemmatized to *synchronize*) to conflate these forms to a single unigram.

Table 5.2: The 2-tuple co-occurrence database with a positive sample sentence. “+” means the sentence comes from the positive dataset.

Sent ID	Term 1	Term 2	Dataset
1	co-administration	Simultaneous	+
1	increase	co-administration	+
1	co-administration	cyclosporine	+
1	increase	significantly	+
1	level	blood	+
1	increase	level	+
1	level	sirolimus	+
⋮	⋮	⋮	⋮

Extended collocation model

By parsing all sentences from the training datasets, we create a syntactic co-occurrence database consisting of 2-tuples as candidates. We propose an extended collocation model to extract bigram cues from the co-occurrence databases. As we mentioned earlier, words in a bigram cue must have both co-existence and discriminativeness properties. Since traditional collocation mining approaches only guarantee the two terms co-exist with each other, we extend these approaches to make cues useful for identifying specific drug interaction relations.

We look beyond simple co-occurrence frequency and evaluate various association measures that

scale it w.r.t. marginal frequencies. We investigate 13 association measures: the baseline co-occurrence frequency from the positive dataset and the 12 association measures from Table 5.5.

The measures from Table 5.5 can be understood in the context of contingency tables [37]. Table 5.3 depicts the traditional contingency table capturing overlaps in occurrences of two terms across sentences. To account for both positive and negative instances of drug interaction sentences, we create two virtual words: "**pos**" and "**neg**", and we make the assumption that any 2-tuple in the positive co-occurrence database automatically collocates with "pos", and that any 2-tuple in the negative database virtually collocates with the term "neg". Thus we 'lift' the traditional bigram contingency table into a trigram table as in Table 5.4. The entries of this table—i.e., observed and expected frequencies—directly feed into the definitions of the association measures (see Table 5.5).

Table 5.3: Traditional contingency table for measuring collocation between w_1 and w_2.

	w_2	$\neg w_2$	$TOTAL$
w_1	O_{11}	O_{12}	R_1
$\neg w_1$	O_{21}	O_{22}	R_2
$TOTAL$	C_1	C_2	N

Table 5.4: Extended contingency table with virtual words "pos" and "neg". For O_{ijk}, *i, j, k* means w_1, w_2 and virtual words, respectively.

	"pos"	*"neg"*	$TOTAL$
$w_1\ w_2$	O_{111}	O_{112}	R_{11}
$w_1 \neg w_2$	O_{121}	O_{122}	R_{12}
$\neg w_1\ w_2$	O_{211}	O_{212}	R_{21}
$\neg w_1 \neg w_2$	O_{221}	O_{222}	R_{22}
$TOTAL$	C_+	C_-	N

5.4.3 N-gram Cues (n≥ 3)

A trigram or other higher order n-gram consists of n words. Although these words are not required to be sequential or next to each other, we require that any two of them are connected by a valid grammatical relation. The process of finding trigrams and other high order *n*-grams is similar to the one described earlier for bigrams. We first construct *n*-tuple co-occurrence databases, recording the *n*-gram candidates and their frequencies in both positive and negative datasets. We then import the same concept of virtual words "pos" and "neg" and change the *n*-gram cue mining problem to an *(n+1)*-gram collocation mining problem. We use association measures to rank all these candidates, and high scoring collocations will be selected as valid cues. While the construction of bigram candidates was relatively straightforward, the construction of *n*-gram cues is more involved.

Table 5.5: Modified association measures for mining bigram cues from the extended trigram contingency table (by adding the virtual words as the third term)

No.	Association Measures
1)	$\chi^2_{Yates}=\sum_{ijk}\frac{(\lvert O_{ijk}-E_{ijk}\rvert-0.5)^2}{E_{ijk}}$
2)	*t-score*$=\frac{O_{111}-E_{111}}{\sqrt{O_{111}}}$
3)	*log-likelihood*$=2\sum_{ijk}O_{ijk}\cdot\log\frac{O_{ijk}}{E_{ijk}}$
4)	*poisson-stirling*$=O_{111}\cdot(\log\frac{O_{111}}{E_{111}}-1)$
5)	*average-MI*$=\sum_{ijk}O_{ijk}\cdot\log_2\frac{O_{ijk}}{E_{ijk}}$
6)	*pointwise-MI*$=\log_2\frac{O_{111}}{E_{111}}$
7)	*local-MI*$=O_{111}\cdot\log_2\frac{O_{111}}{E_{111}}$
8)	*MI*$^2=\log_2\frac{O^2_{111}}{E_{111}}$
9)	*MI*$^3=\log_2\frac{O^3_{111}}{E_{111}}$
10)	*dice*$=\frac{3O_{111}}{(R_{11}+R_{12})+(R_{11}+R_{21})+C_+}$
11)	*jaccard*$=\frac{O_{111}}{N-O_{222}}$
12)	*odds-ratio*$=\log\frac{(O_{111}+0.5)(O_{221}+0.5)(O_{122}+0.5)(O_{212}+0.5)}{(O_{121}+0.5)(O_{211}+0.5)(O_{112}+0.5)(O_{222}+0.5)}$

N-tuple concatenation: We construct *n*-gram candidates based on the original grammatical relations, which consist of only two words in each relation. When two grammatical relations share one word, they can be linked using the shared word as a pivot. This word linkage method is also used in [142] to mine traditional collocations from a single dataset. Without additional restrictions, the number of *n*-tuples using the word linkage method could be prohibitively large. Toward this end, we require that all concatenations be performed in the *same* sentence, i.e., only grammatical relations from the same sentence can be linked. This restriction severely prunes the number of candidate *n*-tuples. We give an example of the concatenated 3-tuples in a co-occurrence database in Table 5.6. The eight 3-tuples are constructed based on the seven 2-tuples in Table 5.2. As we can see, the number of 3-tuples does not increase too much due to the single sentence restriction.

For the *n*-tuple concatenation case, we generalize in the obvious way the 3-tuple example above: *n*-tuples are derived from the lower order *(n-1)*-tuple lists again maintaining the sentence restriction, see Alg. 1. For example, *simultaneous co-administration* in Fig. 5.2 can be extended to the candidates of *simultaneous co-administration cyclosporine* or *simultaneous co-administration increase*. Meanwhile, by introducing the virtual words of "pos" and "neg", the *n*-gram (n$\geq$3) cue mining problem is lifted into a *(n+1)*-gram collocation extraction problem. The contingency table is extended similarly and the association measures from Table 5.5 are similarly generalized. For example, all the association measures in Table 5.5 for the bigram cue mining can be easily extended for the trigram: For the equations 1, 3, 5, we still process all the cells in the contingency table in the same way. For 2, 4, 6, 7, 8 and 9, we only need to replace O_{111} and E_{111} with the observed frequency and expected frequency for the same cells in the trigram contingency table where all the

Table 5.6: The 3-tuple co-occurrence database with a positive sample sentence. "+" means that this sentence comes from the positive dataset, and SID is short for "sentence ID".

SID	Term 1	Term 2	Term 3	Dataset
1	Simultaneous	co-administration	cyclosporine	+
1	Simultaneous	co-administration	increase	+
1	co-administration	cyclosporine	increase	+
1	co-administration	increase	significantly	+
1	increase	significantly	level	+
1	increase	blood	level	+
1	increase	level	sirolimus	+
1	blood	level	sirolimus	+
⋮	⋮	⋮	⋮	⋮

input : A group of *2-tuples* $T_2 =< t_{21}, t_{22}, \ldots, t_{2a} >$
output: $T_3, \ldots, T_n$; $n \geq 3$

for $i \leftarrow 3$ **to** n **do**
    $T = \varnothing$;
    $c = 0$;
    foreach $t_{(i-1)b}$ *in* T_{i-1} **do**
        result = `Expansion(`$t_{(i-1)b}$`);`
        while $\exists\, t \in$ result *and* $t \notin T$ **do**
            $t_{ic} = t$;
            `Add` t_{ic} **to** T;
            c++;
        end
    end
    $T_i = T$;
end

Algorithm 1: Level-wise n-tuple concatenation algorithm.

words co-occur. For Dice and Jaccard in 10 and 11, we apply the same idea from set theory; for 12, a higher order odds-ratio (the ratio of two odds ratios) should be used.

5.5 Text Categorization Using SVM

Finally, as described in our flowchart before, an SVM classifier is used to classify search results from the expanded queries, thus speeding up drug interaction sentence identification. Joachims [73] highlights the many reasons why SVMs are promising algorithms for text classification: 1) the high dimensionality of representation spaces can result in overfitting for some other classifiers; 2) in text

categorization, few features are irrelevant; 3) the feature vectors contain too many zero entries; and 4) most text categorization problems are linearly separable. In our experiments, we use SVMlight[5] with a linear kernel (which has been shown to outperform other kernels [177]).

Most text categorization implementations work with pre-processed input, such as mapping to a bag of words (BOW) representation and weighting the document (sentence) vectors suitably. We opted to not use stemming, since terms could play very different roles in identifying drug interactions even though they might have the same stem [6]. For example, in the sentence

"the use of zolmitriptan in patients receiving MAO-A inhibitors is contraindicated,"

the combination of *use* and *receiving* (as a present participle) strongly hints at being treated with two different drugs simultaneously, but the stem *receiv* which is acquired by stemming a normal verb *receive*, could be rarely used when expressing two drug treatments at the same time. From the training datasets, only 31 sentences containing *receive* in the positive dataset are classified positive (0.7%), whereas 131 sentences have *receiving* as a present participle describing *patient(s)* (3.3%). Conversely, in the negative dataset, 224 instances contain *receive* (4.4%), whereas 21 sentences have *receiving* describing *patient(s)* as a present participle (0.4%). This demonstrates that authors prefer to use the present participle form *receiving* to express drug interaction. For term weighting, we investigate four different methods: binary, term-frequency tf, tf-idf, and tf-rf. All these weighting schemes are similar to the concepts from the traditional information retrieval domain. In the last tf-rf scheme, rf has the same meaning of relative frequency as in Section 5.4.1.

5.6 Experimental Results

5.6.1 System Design and Setup

Data preparation

The DrugBank database contains 4772 drug entries (a.k.a. drug cards) corresponding to more than 12,000 different trade names and synonyms. Among all these drugs, more than 1350 are FDA approved small molecule and biotech drugs, and 3243 are experimental drugs. However, only 1036 of these 4772 entries contain drug interaction descriptions (short paragraphs), so we focus on these drugs.

The 1036 drug entries in DrugBank include valid webpage links in the column *interaction_insert*. We retrieved all of these webpages about detailed drug interactions, extracted plain text using the

[5] http://svmlight.joachims.org/, version 6.02

[6] Stemming is much more aggressive than lemmatization, e.g., *receiving* is changed to *receive* after lemmatization, but to *receiv* after being stemmed. Stemming is usually used when *receiving* and *received* need to be treated as a single form.

Table 5.7: SVM performance with different combinations of weighting schemes and kernels (F:F-measure, P:Precision, R:Recall)

Binary	**Linear**			**Polynomial**		
	F	P	R	F	P	R
1	89.99	95.07	85.44	76.62	95.29	64.08
2	93.33	94.78	91.93	84.14	95.57	75.16
3	92.38	94.68	90.19	83.56	94.6	74.84
4	92.02	94.8	89.4	83.24	96.65	73.1
5	90.42	93.46	87.58	81.88	96.58	71.07
Avg	91.63	94.55	88.90	81.89	95.73	71.65
LOOCV	94.09	95.54	92.7	85.68	96.63	76.96

TF	**Linear**			**Polynomial**		
	F	P	R	F	P	R
1	90.13	95.57	85.28	58.72	93.13	42.88
2	93.06	94.9	91.3	64.37	94.21	48.89
3	92.42	95.14	89.87	66.86	95.04	51.58
4	91.51	94.44	88.77	58.39	95.34	42.09
5	90.11	94.64	86.01	57.49	95.96	41.04
Avg	91.45	94.93	88.24	61.17	94.73	45.29
LOOCV	93.55	95.43	91.75	66.53	95.18	51.14

TF-IDF	**Linear**			**Polynomial (j=1.55)**		
	F	P	R	F	P	R
1	90.07	93.53	86.87	85.28	85.76	84.81
2	93.86	95.72	92.09	70.44	54.56	99.37
3	93.36	95.53	91.3	73.07	58.06	98.58
4	91.82	94.03	89.72	78.37	94.22	67.09
5	91.08	93.98	88.36	78.42	90.53	69.18
Avg	92.04	94.55	89.66	77.12	76.62	83.80
LOOCV	95.08	96.36	93.84	90.21	95.74	85.3

TF-RF	**Linear**			**Polynomial**		
	F	P	R	F	P	R
1	89.2	96.33	83.07	69.76	96.64	54.59
2	92.88	96.11	89.87	75.70	97.51	61.87
3	91.5	95.52	87.82	71.46	96.76	56.65
4	91.08	95.17	87.34	70.75	96.97	55.7
5	89.88	95.09	85.22	65.55	96.92	49.53
Avg	90.91	95.64	86.66	70.65	96.96	55.66
LOOCV	93.48	96.34	90.8	74.31	97.19	60.15

HTML parser tool [7], and segmented them into sentences using the LingPipe[8] toolkit. The total number of the sentences extracted in this manner was 9407. To produce a high-quality training dataset, we created a simple drug name dictionary using DrugBank's *Generic Name*, *Brand Name*, and *Synonyms* fields, and removed all sentences that do not have any entry from this dictionary. This reduced the number of valid positive sentences to 3900. For instance, from the description paragraph for the drug *Nicotine*:

[7] http://htmlparser.sourceforge.net/

[8] http://alias-i.com/lingpipe/

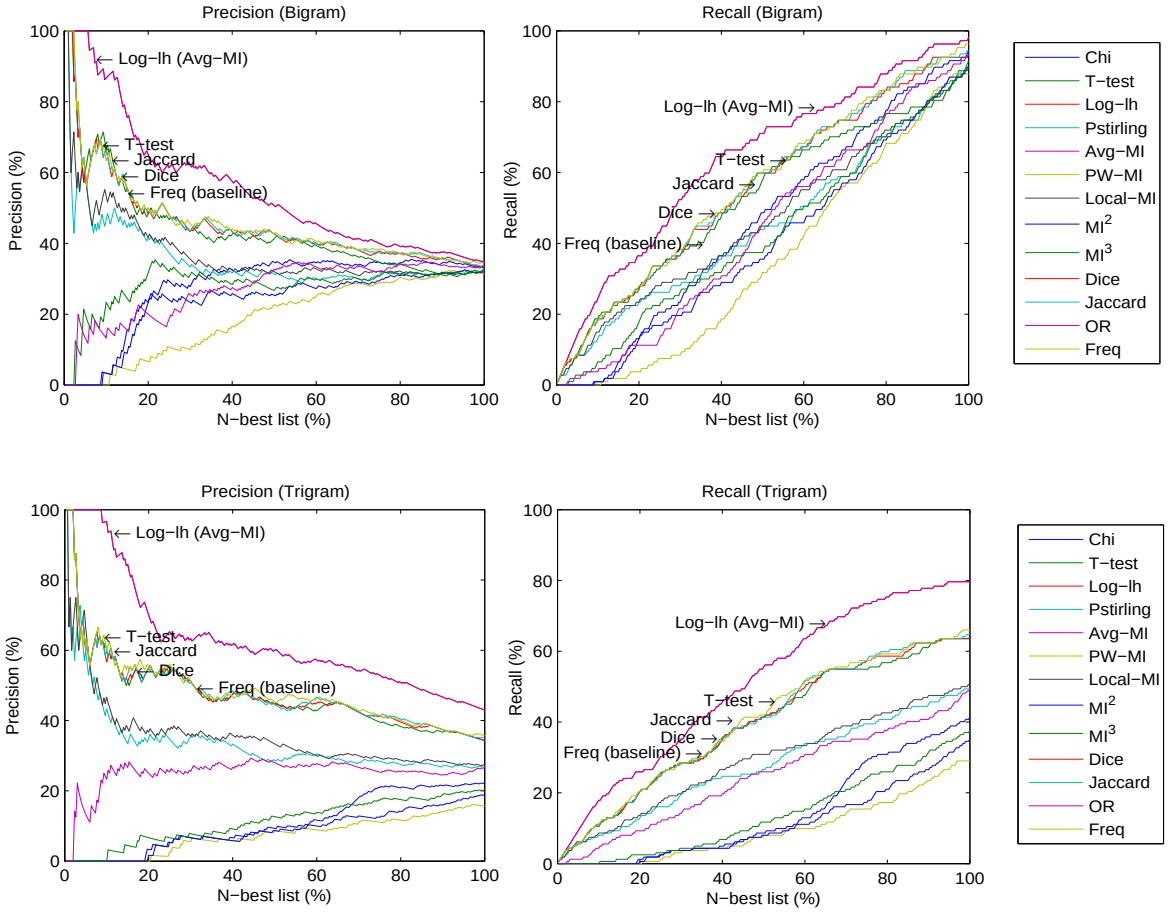


Figure 5.3: N-best evaluation (top 300): comparison of different methods on bigram and trigram by using precision and recall. Log-likelihood and average mutual information perform the best among all measures studied here.

> "Physiological changes resulting from smoking cessation, with or without nicotine replacement, may alter the pharmacokinetics of certain concomitant medications, such as tricyclic antidepressants and theophylline. Doses of these and perhaps other medications may need to be adjusted in patients who successfully quit smoking."

we see that the first sentence is a valid drug interaction sentence but the second one merely gives a warning about required dose adjustment due to the interaction.

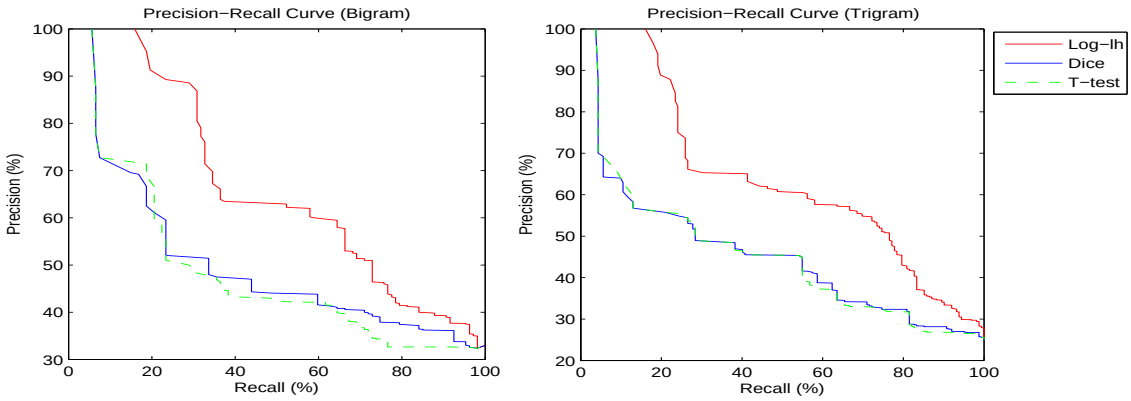


Figure 5.4: Precision-Recall Curve: comparison of different methods on bigram and trigram. Log-likelihood, Dice and T-test methods are listed here.

Table 5.8: Top-20 single-term cues. "tf-pos" and "tf-neg" are the term frequency in the positive and negative datasets, respectively.

Rank	MI			F-score			RF		
	term	tf-pos	tf-neg	term	tf-pos	tf-neg	term	tf-pos	tf-neg
1	concomitant	398	16	concomitant	398	16	coadministration	236	4
2	when	512	131	when	512	131	closely	44	0
3	coadministration	236	4	other	442	138	coadministered	119	3
4	interaction	285	27	interaction	285	27	contraindicated	39	1
5	interactions	247	17	coadministration	236	4	drug-drug	38	1
6	administration	459	210	interactions	247	17	caution	150	4
7	concomitantly	178	6	administration	459	210	exercised	37	0
8	concentrations	329	113	administered	341	123	concurrently	90	3
9	administered	341	123	concentrations	329	113	concomitantly	178	6
10	increase	320	117	concomitantly	178	6	steady	28	0
11	caution	150	4	increase	320	117	co-administered	83	3
12	clearance	173	23	caution	150	4	concomitant	398	16
13	metabolized	136	6	clearance	173	23	adjustments	23	1
14	co-administration	142	9	agents	217	59	metabolized	136	6
15	agents	217	59	co-administration	142	9	careful	20	0
16	warfarin	128	7	metabolized	136	6	warfarin	128	7
17	coadministered	119	3	warfarin	128	7	steady-state	91	5
18	recommended	154	28	coadministered	119	3	depressants	18	0
19	metabolism	164	41	recommended	154	28	digitalis	18	0
20	concurrently	90	3	metabolism	164	41	monitored	100	6

Generating negative examples

For the purpose of evaluation we generate negative examples of drug interaction sentences using both the LPU method and manual annotation. We first obtained 5141 candidate negative sentences

Table 5.9: Some of our identification results (Agreement: drug interaction consistent with the one in DrugBank; Disagreement: results which disagree with the one in Drugbank; Supplement: new drug interaction for the same drug in DrugBank; New: new interaction while DrugBank has an empty entry; In vitro model: a special type of drug interaction.)

Drug pair	Category	Drug interactions in DrugBank	New drug interactions from MEDLINE
Abciximab, ticlopidine	Agreement	These medications have included heparin warfarin, beta-adrenergic receptor blockers, calcium channel antagonists, angiotensin converting enzyme inhibitors, intravenous and oral nitrates, ticlopidine, and aspirin. [DB00054]	concomitant abciximab plus ticlopidine treatment yields a platelet inhibition profile that is a composite of the effects of the 2 agents. [PubMed:10966553]
Alteplase, angiotensin-converting-enzyme inhibitor	Supplement	The interaction of Activase with other cardioactive or cerebroactive drugs has not been studied. In addition to bleeding associated with heparin and vitamin K antagonists, drugs that alter platelet function (such as acetylsalicylic acid, dipyridamole and Abciximab) may increase the risk of bleeding if administered prior to, during, or after Activase therapy. [DB00009]	they warn that patients who are taking an angiotensin-converting-enzyme inhibitor may be at increased risk for angioedema with concomitant alteplase therapy. [PubMed:10813008]
Goserelin, testosterone propionate	In-vitro model	No formal drug-drug interaction studies have been performed. No confirmed interactions have been reported between ZOLADEX and other drugs. [DB00014]	concomitant administration of goserelin or org 30276 and testosterone propionate to castrated rats resulted in a further decrease of the pituitary 5 alpha-reductase activity, compared to the castrated, gnrh-analogue treated rats. [PubMed: 2141376]
Rituximab, fludarabine	In-vitro model	There have been no formal drug interaction studies performed with RITUXAN. However, renal toxicity was seen with this drug in combination with cisplatin in clinical trials. [DB00073]	the study of the effect of fludarabine and rituximab in six freshly isolated b-cell chronic lymphocytic leukaemia (b-cll) samples showed that, in most cases, fludarabine has an additive cytotoxic activity with rituximab and complement. [PubMed: 11564066]
Etanerept, methotrexate	Disagreement	Specific drug interaction studies have not been conducted with ENBREL. However, it was observed that the pharmacokinetics of ENBREL was unaltered by concomitant methotrexate in rheumatoid arthritis patients. [DB00005]	inhibition of articular destruction was also proven by administration of the biologic agents etanercept and infliximab plus methotrexate. [PubMed: 12665969]
Lutropin alfa, follitropin alfa	New	Drug Interactions Not Available. [DB00044]	conclusion: subcutaneous co-administration of 75 iu lutropin alfa with follitropin alfa is safe and effective in inducing follicular development in women with profound gonadotrophin deficiency. [PubMed: 18485121]
Leuprolide, ethanol	New	Drug Interactions Not Available. [DB00007]	further, twice daily administration of leuprolide (50 microg/kg, s.c), concomitant with ethanol, prevented the gradual increase in marble-burying behavior after ethanol-withdrawal and this effect was comparable to fluoxetine (5 mg/kg, i.p.). [PubMed: 18448097]

from the search results by using only drug names. Later, using LPU, 45 sentences were treated as positive and removed automatically; in the manual labeling approach, about 132 positive sentences were removed (2.56% of 5141). Finally we harvested 5096 negative sentences through the LPU method, and 5009 manually labeled negative sentences in total.

Table 5.10: Some of bi-gram cues from extended collocation mining

bi-gram collocations		loglh	avgmi	dice	t-test
concomitant (*amod*)	administration	2911.79	2100.41	.0088	14.27
	treatment	1421.70	1025.54	.0006	3.44
	use	1354.52	977.08	.0042	9.83
	medication	930.61	671.30	.0006	3.69
	therapy	861.60	621.51	.0011	5.06
increase	coadministration (*subj-x*)	2121.22	1530.14	.0009	4.20
recommend		1172.57	845.83	.0004	3.21
decrease		1042.94	752.32	.0007	3.92
result		1002.32	723.02	.0020	6.70
interfere		955.16	689.00	.70	1.30
alter		818.49	590.41	.0002	2.37
administer	when (*advmod*)	2603.77	1878.22	.0074	13.06
coadministered		1822.00	1314.30	.0042	9.88
co-administered		1257.53	907.11	.0018	6.50
take		1102.67	795.41	.0007	3.89
give		1081.87	780.40	.0030	8.33
use		990.09	714.20	.0025	7.22
initiate		746.48	538.48	.0008	4.19
add		687.92	496.23	.0003	2.58

Table 5.11: Some of the tri-gram cues from extended collocation mining. Tri-grams which are just simple extension of bi-gram cues have been removed.

tri-gram collocations			loglh	avgmi	dice	t-test
increase	plasma	concentration	5512.31	3976.29	.0045	9.32
when	administer	increase	5295.64	3819.99	.0004	2.80
decrease	plasma	concentration	4553.77	3284.85	.0014	5.28
when	administer	inhibitor	4496.14	3243.28	.0003	2.43
drug	increase	concentration	4319.32	3115.73	.0003	2.59

Cue mining

As discussed earlier, for identifying single term cues, we utilized three measures as described in Section 5.4.1: mutual information, Fisher kernel and relevance frequency. Table 5.8 shows that all these measures mined single term cues admirably. Since these ranking lists share many of the same terms, especially in the top portions of the lists, we use the union of the top 50 terms from each list as our single-term cues (consisting of 53 cue words after removing stop words and strong domain-related terms). 92.5% of all these single-term cue words are recognized as strong evidence

words for identifying drug interaction during a manual check.

For mining multi-word cues, we explored both basic bigrams and higher order *n*-grams (n≥3). We first parsed all sentences in both positive and negative datasets, and observed a total of 113,577 grammatical dependencies. By filtering out 31 trivial relations (e.g., "det", "auxpass", "cop", etc.; most of them either are related with stop-words like "the", or create duplicate connections), and removing all the bigrams which appear in the positive dataset less than 10 times, we retained 83417 of them to construct 2-tuple co-occurrence information. Next, with the concatenation idea from Section 5.4.3, we retrieved 167,302 valid records in the 3-tuple co-occurrence database. However, by requiring that valid 3-gram cues must appear in the positive dataset more than 5 times, the number of valid 3-tuples in the candidate co-occurrence database is dramatically reduced to 644. Finally, the 12 association measures from Table 5.5, together with the baseline measure of co-occurrence frequency in the positive dataset, were calculated for both the 2-tuple and 3-tuple co-occurrence databases. Some of the ranked bigram and trigram results are listed in Table 5.10 and 5.11, respectively. Further experiments revealed that mining higher order *n*-gram (n≥4) cues are not necessary for drug interaction search, since all of them are just simple extensions of tri-gram cues.

SVM classification

We compared the classification capability of eight SVM models with different combinations of two kernels (linear and polynomial) and four weighting schemes (binary, tf, tf-idf, tf-rf). We evaluated these models by using *F-measure*, given by

$$F = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

We also conducted a *5-fold cross validation* and a *leave one out cross validation* (LOOCV). Table 5.7 summarizes the detailed model evaluation results. We observe that the linear kernels with binary weighting and tf-idf weighting provide the best performance.

5.6.2 Evaluation

We perform three categories of evaluation. We evaluated the effect of our new collocation mining approach, and also we tested the quality of the training data labeled by the LPU method. We also listed some representative findings from MEDLINE.

N-best evaluation

To evaluate our multi-word collocation mining approaches, we employed the *n*-best evaluation method, which is widely used for evaluating collocation measures [37, 38, 105, 122, 139]. We first

scanned all the *n*-tuples satisfying the threshold requirement (frequency in positive dataset greater than 5), and generated a human-labeled set of real cues as C_{real}. For each association measure, we considered the *n*-tuples from the ranking lists as *valid* only if they also appear in set C_{real}. Obviously, good association measures will always put the real valid cues near the top of their ranking lists. Therefore, after scanning the top-n results in the ranking lists, we draw *precision-recall* graphs for these measures, by calculating the proportion of valid cues in the *n*-best list (*precision*) and the fraction of the number of valid cues from the *n*-best list to the size of C_{real}. Fig. 5.3 shows the precision and recall graphs using the top 300 lists from each measure. The *x*-axis represents the proportion of the ranking list, while the *y*-axis depicts the corresponding precision (recall) values. The *n*-best evaluation shows that log-likelihood (Log-lh) and average mutual information (Avg-MI) perform significantly better than other measures from the precision-recall graphs (they are the top most curves), a result consistent with prior research [36, 105]. Measures such as the Dice coefficient, Jaccard and t-test perform comparably to the baseline approach of positive frequency (Freq), while the other measures perform poorly. For the rest of this chapter, we choose the log-likelihood method as our association measure. Fig. 5.4 shows the precision-recall curves for Log-lh, Dice, and T-test.

Evaluating the quality of LPU labeling

Our experiments also show that the negative dataset generated automatically by the LPU is comparable to the one labeled manually, and that the difference between these two methods for constructing negative datasets has little influence on the final results of cue word mining. We compared the *average precision* values and also the *normalized discounted cumulative gain* (nDCG) values [29, 70] of the multi-word cue mining results from these two different negative datasets. Average precision is calculated by averaging the precision values from the rank positions where a valid cue is retrieved, and the nDCG value for the top-p list is calculated as $nDCG_p = \frac{DCG_p}{IDCG}$. Here the DCG is defined as:

$$DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i}$$

where rel_i is 1 when the *i*-th *n*-gram in the list is judged as a valid cue, and 0 otherwise. IDCG means the possible maximum DCG value when all the valid cues are ranked at the top [70].

Table 5.12 shows that there is no significant decrease of average precision or nDCG values when we use the LPU method to generate the negative dataset. In fact, the final ranking lists (top 100) based on these two different negative datasets share many cue words. For the bigram approach, 96 of the top 100 cues are identical, and for the trigram approach, 97 of the top 100 are identical.

Representative results

Table 5.9 depicts some of the results of our analysis, which classifies all inferred drug interaction sentences into five categories. Most of the results are consistent with the original drug interaction

Table 5.12: Collocation mining effects of log-likelihood method by using different negative datasets (top 100 ranking lists)

Method	Bigram		Trigram	
	AvgP (%)	nDCG	AvgP (%)	nDCG
Manually labeling	83.84	0.967	85.50	0.971
LPU	80.34	0.958	84.07	0.968

descriptions in DrugBank (same drug, same interaction). Some describe additional interactions (i.e., same drug, but interactions with new drugs). We also found some interaction description sentences which disagree with the ones in DrugBank. Among the 4772 entries in DrugBank, most of them (3736) have empty drug interaction records, and we found new interaction information for some of these drugs. Finally, our approach also identifies drug interactions from in-vitro studies, a category often ignored in clinical studies.

5.7 Conclusions and Future work

We have introduced a framework for mining data-driven linguistical cues and for using these cues to aid querying for drug interactions. Our framework integrates traditional collocation mining with discriminative association with positive/negative training instances. Compared to other systems which require *a priori* domain knowledge, our system seeks to solve this problem with the help of linguistic features alone. By identifying many drug interactions not currently curated in DrugBank, our experimental results demonstrate that this is a promising approach.

Since our framework uses only linguistic cue words to augment queries, this approach can be re-targeted toward many other domain-specific needs, such as for modeling biochemical interactions and subjective opinions. This is one direction of future work. A second issue we are investigating is to develop more structured, graph-theoretic, representations of linguistic cues from dependency parses, i.e., beyond simple sets of words as used in collocations.

Chapter 6

Modeling virtual ratings for recommender systems

6.1 Introduction

Collaborative filtering has now become established in commercial applications, e.g., Amazon [91] and been improved over the years by many researchers [5, 10, 33, 67, 79, 91, 120, 126, 136, 158]. Many of these collaborative filtering methods are based on explicit preference information (e.g. rating) although implicit feedback methods such as a user's purchase history or browsing history have also been studied [66]. Unifying collaborative and content-based filtering has been demonstrated to improve recommendation accuracy [6]. Many evaluation metrics have been proposed [22, 23, 62]. Contemporary surveys of collaborative filtering algorithms can be found in, e.g., [1, 150].

A key problem in collaborative filtering is the continuing need for better information about users' past history, generally referred to as the *missing rating* problem [100, 101]. Here, we propose a framework to utilize unstructured review comments from users to generate *virtual ratings* and, in turn, improve recommendation performance. Unlike the *rate it again* strategy of [3], our work introduces virtual ratings to help bridge the gap between unstructured text and traditional rating-based methods. Although our methods can be adapted toward many collaborative filtering algorithms, we adopt the item-based algorithm [136] as baseline and demonstrate how our approach improves upon this baseline.

In our study, we utilize ten years of publicly available large scale movie review comments and rating information from the IMDb website. As shown in Figure 6.1, there is a significant amount of unrated review comments from Jan. 1999 to Oct. 2009 in IMDb. Traditional collaborative filtering algorithms will focus only on the explicit rating information, and discard the textual reviews. We propose a *read and rate* strategy to utilize review text by generating virtual ratings. While enriching the rating information, our method is also compatible with all rating-based conventional collaborative filtering algorithms. Our experimental results show that incorporating the comments

helps predict the future ratings, especially for those users who do not have a long rating history.

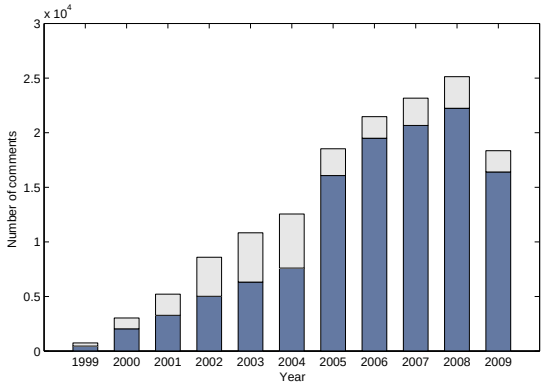


Figure 6.1: Ratio of unrated comments to rated comments from Jan. 1999 to Oct. 2009 in IMDb.com by loyal users who rate more than 10 movies per year on average.

Our primary contributions are:

1. We present a novel, iterative, strategy to enrich rating data by textual analysis of review comments, and also a global topic analysis method. Experimental results demonstrate that the iterative strategy quickly converges to generate virtual ratings, and the topic model based method gives the best performance. Our *read and rate* method inherently overcomes users' sparse rating history and can hence improve prediction accuracy. Significantly, our approach can be combined with any rating-based collaborative filtering technique and any domain where (unrated) comments are available.

2. We propose to use both single-term and dependency features inferred from text to generate the virtual ratings. We show that dependency features can provide significant discriminatory information about user sentiments.

3. We introduce user-based evaluation metrics as variants of the traditional MAE and RMSE measures, to help assess our experimental effectiveness.

6.2 Methods

Our approach broadly comprises of two stages. We first generate virtual ratings for those users (or movies) only having few real ratings, and subsequently use these virtual ratings along with the original ratings together in a conventional collaborative filtering algorithm (in this work, we use

the popular item-based approach). We focus on the extraction of these virtual ratings from text, and their accuracy.

As shown in Figure 6.2, in IMDb, users are required to write short paragraphs for the movies when they submit their comments. However, the ratings are not mandatory, and hence for some comments there is no corresponding rating information. Each row in Figure 6.2 shows a specific user's rating and comment history, while the columns represent the movies. Since not all the comments are provided with rating information, our task is to generate virtual ratings for enriching the user's rating history and in turn to improve the accuracy of the item-item similarity matrix.

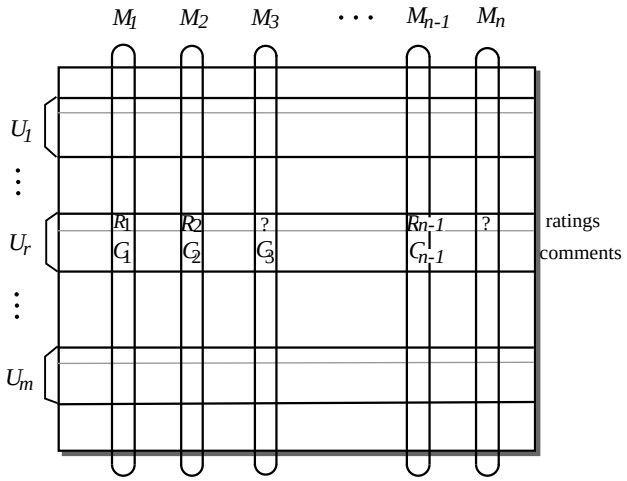


Figure 6.2: Problem setup: $U_1, \ldots, U_m$ are the users, $M_1, \ldots, M_n$ are the movies, R denotes the rating values, and C represents comments. Question marks denote incomplete ratings.

In the following subsections, we introduce both the iterative strategy and the global topic analysis method (topic model based) for generating virtual ratings, and we also analyze the power of linguistic dependency as an important textual feature.

6.2.1 Domain-specific sentiment Feature Extraction

We use discriminative sentiment features in the first iterative strategy. To extract these sentiment textual features, we study both single-term and the bi-gram dependency approaches. We setup supervised learning scenarios based on user-item entries that have both review comments and ratings. For these entries, the review text is classified into two categories: positive and negative. We assume that those comments with a rating equal to or greater than 7 stars can be treated as positive, while comments with a rating equal to or less than 4 stars are treated as negative. We then evaluate the ability of different feature extraction methods to discriminatively distinguish between these classes.

Single Term Extraction

We utilize three criteria functions to rank all the tokens/terms from the whole dataset.

Mutual information (MI): Mutual information is widely used as a means for feature selection [152] in text categorization. In our study, we do not use it as a simple feature reduction method, but rather as a criteria function to rank all the terms according to their dependency on the category label (positive review or negative review). When a term t_i appears with a highly biased presence in one category *c* (good movie or bad movie in our study) more than the other, we expect it to have sufficient discriminatory power. The mutual information between t_i and *c* is given by:

$$MI\left(t_i, c\right) = \sum_{t_i \in \{0,1\}} \sum_{c \in \{+,-\}} P\left(t_i, c\right) \log \frac{P\left(t_i, c\right)}{P\left(t_i\right) P\left(c\right)}$$

Fisher kernel: We use a variant of the Fisher kernel [24]. The *F-score* for the *i*-th term is defined as:

$$F\left(i\right) = \frac{\left(\overline{x}_i^{(+)} - \overline{x}_i\right)^2 + (\overline{x}_i^{(-)} - \overline{x}_i)^2}{\frac{1}{n_+ - 1} \sum_{j=1}^{n_+} \left(x_{j,i}^{(+)} - \overline{x}_i^{(+)}\right)^2 + \frac{1}{n_- - 1} \sum_{j=1}^{n_-} \left(x_{j,i}^{(-)} - \overline{x}_i^{(-)}\right)^2}$$

Here, $x_{j,i}^{(+)}$ and $x_{j,i}^{(-)}$ represent the *i*-th feature values in the *j*-th review comments in the positive and negative datasets, respectively. n_+ and n_- are the number of positive and negative review comments from the training dataset, $\overline{x}_i$, $\overline{x}_i^{(+)}$, $\overline{x}_i^{(-)}$ are the average weight values of the *i*-th terms computed across the whole dataset, the positive review dataset, and the negative review dataset, respectively. In our study, a simple binary weighting scheme is used: the weight for a term is 1 if it appears in the review comment, 0 otherwise.

Support Vectors: Our final ranking criteria comes from the SVM feature selection literature. As is well known, the linear model obtained from the SVM training process can be used to calculate the relevance of the hyperplane features [24, 57]. We calculate the weight vector of the hyperplane from the trained linear model, and use these weight values to rank terms according to their importance. Given the labeled training dataset (x_i, y_i),$x_i \in R^n$,$y_i \in \{1, -1\}$, the SVM optimization problem is [161]:

$$\min_{w,b} \quad \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi(w, b; x_i, y_i)$$

Here $|w_j|$ is the weight value we use to rank the *j*-th term.

Dependency Extraction

When no single term in a sentence can help to clearly identify the polarity of the review text, multi-word grammatical structure may help. Consier the three example sentences below: the com-

bination of “supposed” and “funny” indicates strong evidence of a negative review.

> (1) “Jar Jar didn’t get a single laugh from the audience through the film, yet he’s ***supposed*** to be ***funny***?!?”
>
> (2) “He is so much the definition of something that was ***supposed*** to be ***funny*** but ended up so not being.”
>
> (3) “In this movie, it’s ***supposed*** to be ***funny*** that an office building has a seven-and-a-halfed floor, that an apartment is filled with animals (hello, Ace Ventura anyone?)”

We utilize a dependency parsing tool to break down sentences and treat grammatical relations as independent ‘virtual words’ to be added to the initial single-term feature set. We choose the Stanford typed dependency parser [1] [102] as our parsing tool in our experiment. The collapsed typed dependencies (a.k.a. grammatical relations) are used to form the virtual words. To simplify the computation process, we only retain dependencies whose terms have the part-of-speech tags such as “*NN*", “*VB*”, “*JJ*”, “*PRP*” and “*RB*”[2].

Figure 6.3 shows a typical parsing result from one sample sentence. The grammatical relations are marked along the edges.

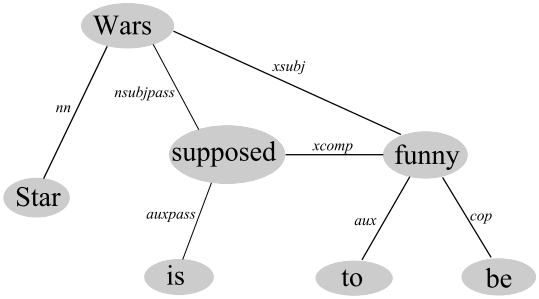


Figure 6.3: Dependency parse of the sentence “Star Wars is supposed to be funny”. The edge label shows the grammatical relation name.

We use discriminative dependencies as textual features to do text analysis, and we adopt the supervised collocation mining method from [56] to retrieve discriminative dependencies. The goal is to retrieve dependencies which satisfy that their left and right term collocate with each other a lot, and also ensure that the whole dependencies tend to exist in only one kind of review, either positive or negative (but not both). We use the log-likelihood and average mutual information metrics to rank the dependencies:

[1] http://nlp.stanford.edu/software, V1.6

[2] In the real implementation, we considered all the POS tags with these five tags as prefix, such as “*NNS*”, “*VBN*”, etc.

$$\text{log-likelihood} = 2 \sum_{ijk} O_{ijk} \cdot \log \frac{O_{ijk}}{E_{ijk}}$$

$$\text{average-MI} = \sum_{ijk} O_{ijk} \cdot \log_2 \frac{O_{ijk}}{E_{ijk}}$$

Here i and j represent the left term and right term, respectively. k denotes the category, either positive (value is 1) or negative (value is 0). O_{ijk} is the observed frequency when left term i and right term j form a dependency and belong to the category k. E_{ijk} denotes the expected frequency.

6.2.2 Virtual Rating Generation

As in Figure 6.2, our task is to make the rating matrix denser by filling as many incomplete ratings as possible in an accurate way. Our virtual rating generation is close to the opinion mining and sentiment analysis in natural language processing field [51, 106, 108, 119, 129]. However, we currently do not exploit other deep sentiment analysis methods, such as comparative sentence analysis or relation extraction. Many other complicated and time-consuming NLP processes such as POS-tagging, named entity recognition or coreference resolution are not considered either. Unlike the framework in [89], we do not use hybrid rules, and our method focuses on the usage of single term and multi-word (dependency) linguistic features. We propose three straightforward methods to generate the virtual ratings in an iterative way: SVM regression, a weighted sum approach, and a variant of k-nearest neighbor. We also propose a global analysis method, which is based on LDA (Latent Dirichlet Allocation).

Iterative Methods

SVM regression: We treat the issue of uncovering the incomplete ratings as a regression problem. The dependent variable is the rating value and the independent variables are the features extracted from comments. By analyzing the relationship between comments and their known ratings, regression helps to predict the unknown rating values from the unrated comments. We adopt SVM-Light as the regression tool in our study.

Weighted Sum: To predict virtual ratings from comments, we also utilize the similarity between the comments' text. The virtual rating r_i from the review text R_i is calculated by the following formulas, where r_n is the known rating for a specific user or movie, and R_n is the corresponding review text to r_n.

$$r_i = \frac{\sum_{n \in N} r_n sim(R_i, R_n)}{\sum_{n \in N} |sim(R_i, R_n)|}$$

We use the Jaccard coefficient as the similarity measure of choice. If C_i and C_n are the collections of features for review text R_i and R_n, then:

$$sim(R_i, R_n) = \frac{|C_i \cap C_n|}{|C_i \cup C_n|}$$

kNN Variant: Finally, we also consider a k-nearest neighbor approach. A traditional kNN regression calculates values by averaging all the k nearest neighbors' values. However, the virtual ratings are not only required to be close to the rating values of the similar text, but also to be different from the rating values of dissimilar text. We propose a variant of the traditional kNN algorithm for this purpose, wherein we utilize both the k nearest neighbors and the k farthest neighbors:

$$r_i = \arg\min_r \sum_{n \in N} \frac{|r - r_n|}{1 - sim(R_i, R_n)}$$

The similarity is again measured by Jaccard coefficient, and N is the union of the k nearest neighbors and the k farthest neighbors.

Algorithm 2 depicts the approach we use to generate virtual ratings. As shown, it is an iterative process. We first generate the ratings for users. Since specific users usually have consistent voice when writing the comments, we can use their known rated comments to form the virtual ratings from unrated comments. After this step, we generate the ratings for movies, considering that comments of a specific movie normally share lots of contextual words. Then we repeat the process above until no new virtual ratings can be generated. By using this aggressive approach we fill the incomplete entries in an very efficient manner.

LDA-based Method

In our second strategy, we generate the virtual rating through a global analysis. In the iterative way above, the virtual ratings are generated from the iteratively enriched training data. In this LDA-based method, we only do the topic analysis on the training data just once, and all the virtual ratings are computed through the model. We use not only "Bag of words" for feeding the LDA analysis, but also the mixture of "BOW" and dependencies. We require that dependencies for LDA must have specific pos tags, such as "NN", "VB", "JJ", etc. We adopt the predictive perplexity as our measure for virtual rating calculation.

Predictive perplexity: LDA-based predictive perplexity (PP) is originally mentioned as a way to do collaborative filtering [18]. The predictive perplexity is calculated as:

$$PP(D_{test}) = exp\{-\frac{\sum_{d=1}^{M} \log p(w_{d,N_d}|\mathbf{W}_{d,1:N_d-1})}{M}\}$$

It predicts each user d's favorite movie w_{d,N_d} given this user's other favorite movies $\mathbf{W}_{d,1:N_d-1}$. M denotes the number of users;

input : M_r, a sparse matrix of rating;
M_c, a matrix of review comments;
Vec_{user}, a user list;
Vec_{mov}, a movie list;
output: M_r, the updated rating matrix;

int updated $=$ MAX_INT;
int threshold $=$ 20;
while updated *>0* **do**
  updated = 0;
  foreach *user* u *in* Vec_{user} **do**
    get u's <rated, comment> pairs $List_1$ from M_r and M_c as : <r_1, c_1>, <r_2, c_2>, ..., <r_m, c_m>;
    get u's unrated comments $List_2$ as: c_1, c_2, ..., c_n;
    if *size of* $List_1$ > threshold **then**
      updated += `VirtualGen(`$List_1$, $List_2$`);`
    end
  end
  foreach *movie* m *in* Vec_{mov} **do**
    get m's <rated, comment> pairs $List_1$ from M_r and M_c as : <r_1, c_1>, <r_2, c_2>, ..., <r_m, c_m>;
    get m's unrated comments $List_2$ as: c_1, c_2, ..., c_n;
    if *size of* $List_1$ > threshold **then**
      updated += `VirtualGen(`$List_1$, $List_2$`);`
    end
  end
end

Algorithm 2: Iterative virtual rating generation algorithm. Function "VirtualGen" is used to assign virtual rating values to the unrated comments given the rated comments. The return value of this function is the count of how many virtual ratings are generated. 20 rated comments are required to ensure the quality of virtual ratings.

In our case, we view documents as analogous to users, while words inside documents as analogous to movies. With a group of words known, we then predict the possibility of another group of unknown words. We choose comments with very small rating values (<=4) as negative training data, and comments with big ratings (>=7) as positive training data. We train the LDA model based on all these comments, and then based on the model, for each specific testing comment, we calculate both the predictive perplexity from the positive training data and the predictive perplexity from the negative training data.

$$PP_{pos}(test, \mathbf{w}_{pos}) = exp\{-\frac{1}{M_{Test}} \sum_{d=1}^{M_{test}} \log \frac{\sum_{i=1}^{m} p(w_d|\mathbf{w}_{pos_i})}{m}\}$$

$$PP_{neg}(test, \mathbf{w}_{neg}) = exp\{-\frac{1}{M_{Test}} \sum_{d=1}^{M_{test}} \log \frac{\sum_{i=1}^{n} p(w_d|\mathbf{w}_{neg_i})}{n}\}$$

PP_{pos} and PP_{neg} represent the likelihood to generate the specific testing comment from the training documents (positive and negative, respectively). $p(w_d|\mathbf{w}_{pos})$ and $p(w_d|\mathbf{w}_{neg})$ are the probabilities to generate the word (w_d) from a single training comment **w** (either from a positive comment $\mathbf{w}_{pos}$ or a negative one $\mathbf{w}_{neg}$). Since we calculate the PP metric from a collection of comments to one document, we average the probabilities in the equations above. m and n are the number of documents in the positive and negative datasets, respectively, M_{test} is the number of words from the testing document. The probability is calculated after the LDA model is estimated by Gibbs sampling:

$$p(w|\mathbf{w}_{obs}) = \int \sum_{z} p(w|z)p(z|\theta)p(\theta|\mathbf{w}_{obs})d\theta$$

Virtual Rating Generation: After observing the predictive perplexity values both from the positive dataset and the negative dataset, we go through a kNN (k=5 in our study) style virtual rating assignment process. We calculate PP_{pos} and PP_{neg} for each comment (both rated and unrated), and by treating PP_{pos} and PP_{neg} as coordinates, we define the distance between any two comments c_1 and c_2 using their distance in two dimensional space. Thereby for each unrated comment, we can find its k-nearest neighbors, which already have their rating values.

$$dist(c_1, c_2) = \sqrt{(PP_{pos_1} - PP_{pos_2})^2 + (PP_{neg_1} - PP_{neg_2})^2}$$

We average these neighbors' ratings as the testing comment's virtual rating.

6.2.3 Rating Prediction

After the virtual ratings are generated, we simply employ traditional collaborative filtering approaches to predict future ratings. We employ an item-based approach: the algorithm accepts a rating matrix with rows as users, column as movies, and predicts the rating values that a specific user would give for a certain movie. It is well-known that the item-based algorithm has more advantages than the user-based ones [91, 136], since the relationship between the items is relatively static. An item-based method rates an unrated movie according to its similarity to those rated movies. In our study, we also use the item-based strategy.

Similarity Matrix Construction: when computing the item similarity matrix, the common and practical way of implementation is to calculate the similarity between two movies only when these

two movies have been commented or rated by at least one common user. In our study we use the adjusted cosine similarity between two movies:

$$sim(m_i, m_j) = \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_{u \in U}(r_{u,j} - \bar{r}_u)^2}}$$

Here $\bar{r}_u$ is the user u's average rating and $r_{u,i}$ is user u's rating value on movie m_i.

Rating Prediction: the predicted rating $r_{u,k}$ for user u of a movie k is computed by using the weighted sum method as below,

$$r_{u,k} = \frac{\sum_{n \in H} r_n sim(m_k, m_n)}{\sum_{n \in H} |sim(m_k, m_n)|}$$

Here H is the collection of u's rated movies and r_n is the history rating from u for movie m_n. The final predicted rating values are normalized to the scale of $[1, 10]$.

Note that users' virtual ratings can be used both in similarity matrix construction and rating prediction. We discuss this in detail in the later part of Section 6.3.5.

6.3 Experimental Results

6.3.1 System Setup

Our evaluation dataset comes from real life IMDb rating and movie comment data. As a popular movie online community, the IMDb website has a vast amount of loyal users (more than 403,226 users in our dataset) and is rich in movie comments (more than 969,891). We collected more than ten years worth of rating and movie comment history (from Jan. 1999 to Oct. 2009), and the ratio of unrated comments to rated comments is shown in Figure 6.1. Our task is to quantify the benefit from generating virtual ratings, both for recommendation capabilities and for recommendation accuracy.

We split the original ten years worth of data into six datasets for our experiments by choosing different time boundaries (6 snapshots). As in Table 6.1, all the rating and comment information before the time boundaries are used as training input, and those ratings after the time boundaries are treated as testing data. However, if a user did not appear in the training data, we do not consider his ratings after the time boundary as valid testing data, and this rule applies to the movies as well.

6.3.2 Feature Extraction Results

As mentioned earlier, we utilize the sentiment key features to analyze the similarity between the review comments. We assume that for a specific domain, the sentiment key terms/dependencies

Table 6.1: Time boundaries, the number of known ratings, and number of ratings to be evaluated for six datasets.

No.	start	end	train_size	test_size
(1)	1/1/1999	1/1/2001	2486	2116
(2)	1/1/1999	1/1/2002	5757	3514
(3)	1/1/1999	1/1/2003	10772	5831
(4)	1/1/1999	1/1/2004	17090	9156
(5)	1/1/1999	1/1/2005	24684	14510
(6)	1/1/1999	1/1/2006	40760	19620

tend to co-occur with specific sentiment topics, and thus those features can be treated as an external textual resource to identify the sentiment.

The single-term key terms are ranked by three methods, and Table 6.2 depicts the top-15 single-term feature extraction results from these three methods. Note that although top-15 single-terms from SVM are quite different from MI and F-score, these three methods all cover the majority of the discriminative terms in top-600. In our experiment, we use the top 1000 positive and top 1000 negative discriminative key terms from each method for safety. The union of all these terms makes the final key term list consisting of 3010 words.

Table 6.2: Single term feature extraction results (top-15 lists)

MI		SVM		F-Score	
+	-	+	-	+	-
great	bad	haters	toxic	great	bad
excellent	worst	wii	predalien	excellent	worst
amazing	waste	excellent	avp	amazing	waste
love	awful	awesome	shinzon	love	awful
perfect	boring	refreshing	dumberer	perfect	boring
loved	terrible	flawless	freaking	loved	terrible
wonderful	stupid	anamorphic	carbon	wonderful	stupid
life	horrible	pleasantly	dryer	life	money
brilliant	money	riveted	exaggerating	brilliant	horrible
story	crap	detractors	backside	story	poor
fantastic	poor	blu	slaps	beautiful	crap
beautiful	worse	complaint	ankle	fantastic	worse
don	minutes	esquire	scenarist	don	minutes
awesome	plot	bravo	whistler	awesome	plot
batman	poorly	mcclane	chop	batman	poorly

Table 6.3 depicts the results of dependency feature extraction (bi-gram feature extraction) discussed in Section 6.2.1. Observe that many of these dependencies involve single-term key terms extracted in the previous section. However, some of them are very interesting, such as “first time”, “real life”, “high hopes”, “blah blah”, where neither the left term nor the right term of these dependencies can easily help to identify the sentiment polarity, but the combination makes them discriminatory.

Table 6.3: Dependency feature extraction results

pos(+)					neg(-)				
Relation	l_term	r_term	loglh	avgMI	Relation	l_term	r_term	loglh	avgMI
amod	effects	special	229441	165507	prep_of	waste	time	75474	54443
advmod	seen	ever	97882.1	70607	dobj		money	68637.9	49511.8
advmod	well	very	93230.4	67251.5	amod		complete	44871.9	32368.2
dobj	did	job	87239.5	62930	nsubj	bad	acting	29452.2	21245.3
nn	Jackson	Peter	72944.2	52618.2	nsubj		plot	29033.6	20943.3
nsubj	see	you	58859.7	42458.3	nsubj		performance	28703.1	20704.9
amod	time	long	56041.8	40425.6	nsubj		ending	24600.2	17745.3
nn	development	character	51681.1	37280.1	dep	blah	blah	15772.3	11377.3
amod	screen	big	51647.8	37256	amod	hour	first	14229.4	10264.4
advmod	again	once	50503.4	36430.5	advmod	nothing	absolutely	13060.1	9420.84
amod	time	first	40238.2	29025.8	dep		just	13006.2	9382
advmod	recommend	highly	38514.4	27782.3	amod		new	12247.8	8834.94
amod	movie	best	30095.1	21709	amod	potential	great	12034.4	8681
amod	film		29297.8	21133.9	dep	make	nothing	12002.8	8658.16
amod	films		27466.9	19813.2	xcomp	supposed	funny	11854.9	8551.53
amod	life	real	24106.9	17389.5	amod	hopes	high	11853.9	8550.75
amod	fan	huge	23409.1	16886.1	dobj	paying	money	11851	8548.7

We also studied the stability of the key term/dependency features. As shown in Figure 6.4, we extract 9 different feature sets from 9 different comment datasets (all the review comments with rating$\geq$7 or $\leq$4). All these datasets have the same starting date on Jan. 1999, but have different ending dates. We extract the feature sets from those datasets, and compare the overlap rate with the feature set extracted from the whole review comments. We found that the overlap rate becomes higher when the dataset gets larger, showing that the domain specific sentiment features tend to be consistent and fixed when we have a large amount of review comments. Notice that the steady increase of overlap rate indicates the decline trend of the variability of the feature set. In other words, if features do not tend to be stabilized, the change of overlap rate should be random. This is an important result since it suggests that our results might be long-lasting as the community around a recommender system grows. However, we also analyze the situation when all the features are generated from the training dataset only (see Section 6.3.6).

6.3.3 Virtual Rating Results

Since our study focuses on the effect of linguistic features on linking the free text and rating value, we use the simple conventional item-based collaborative filtering algorithm as our baseline. Table 6.4 shows the iteration numbers for different datasets (snapshots) and approaches when adopting an iterative strategy. In most situations, these algorithms converge quickly and generate all the

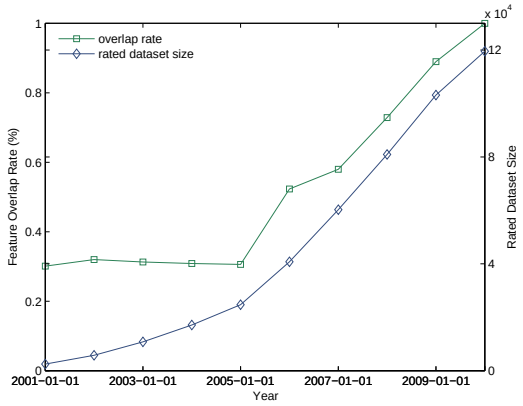


Figure 6.4: Relationship between the overlap rate and the rated dataset size.

possible virtual ratings.

Table 6.4: Number of iterations to generate all the virtual ratings using different methods and datasets)

Method	Datasets					
	(1)	(2)	(3)	(4)	(5)	(6)
SVM	3	4	4	4	4	4
kNN	3	4	4	4	4	4
W-SUM	3	4	4	4	4	4

In Table 6.5, we list the change of the training dataset size. Since we only generate the virtual ratings among training data, the number of testing ratings remains the same for both the baseline and our approaches. For most datasets, the number of generated virtual ratings is about 30%∼50% of the original training size. We notice that the LDA method generates the most virtual ratings among all the methods, since there is no iteration process for LDA method.

6.3.4 Recommendation Performance

Since our experimental datasets are constructed by setting up the time boundaries in a snapshot style, we use the real life future ratings as our evaluation datasets. The testing ratings satisfy the requirement that both the user and the movie of the ratings have to be in the training dataset before. We hence do not conduct evaluations on the ratings for new users or new movies.

While there are many evaluation measures proposed for recommender systems, we adopt the widely-used *mean absolute error* (MAE) and *root mean squared error* (RMSE) as our initial met-

Table 6.5: Training size change after generating virtual ratings (testing dataset size remains constant)

Method	Dataset					
	(1)	(2)	(3)	(4)	(5)	(6)
original_train	2486	5757	10772	17090	24684	40760
SVM_train	3158	8590	17206	27939	40572	59378
kNN_train	3060	8306	16632	27549	40250	59127
W-SUM_train	3073	8319	17064	27744	40477	59310
LDA_train	3772	8982	17572	28400	40956	59483
Test_size	2116	3514	5831	9156	14510	19620

rics. As in the following formulas, $p_{u,j}$ is the predicted rating for user u on movie j, while $r_{u,j}$ is the real rating. N is the number of total testing ratings.

$$\text{MAE} = \frac{\sum_{\{u,j\}} |p_{u,j} - r_{u,j}|}{N}$$

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{\{u,j\}} (p_{u,j} - r_{u,j})^2}$$

However, the general MAE or RMSE is the average value of all the testing ratings. To assess the factors which have influence on the change of MAE or RMSE, we propose the user-based version (U-MAE, U-RMSE) as follows:

$$\text{U-MAE} = \frac{\sum_{u=1}^{N_{user}} \frac{\sum_{j \in T_u} |p_{u,j} - r_{u,j}|}{N_{T_u}}}{N_{user}}$$

$$\text{U-RMSE} = \frac{\sum_{u=1}^{N_{user}} \sqrt{\frac{1}{N_{T_u}} \sum_{j \in T_u} (p_{u,j} - r_{u,j})^2}}{N_{user}}$$

The user-based MAE and RMSE calculate each user's MAE or RMSE value, and then average them by the number of users. N_{T_u} is the number of total testing ratings of user u and N_{user} is the number of users. We take this strategy simply because in a large dataset, although there are many users, the number of users with short rating history could be largely different from the number of users with long rating history. We desire to assess the effect of the virtual ratings on different users with different lengths of rating history.

Figure 6.5 shows the experimental results based on different input rating matrices. The baseline result has no virtual rating, while the other results are based on rating matrices with virtual ratings. These virtual ratings are generated by different approaches using both single term and dependency

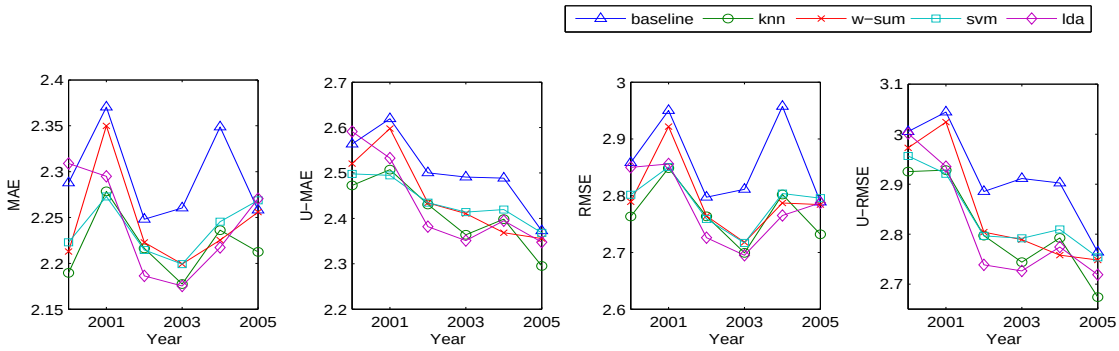


Figure 6.5: Comparison of four proposed virtual rating generating methods with the baseline. Different evaluation metrics are used, and all these methods use both single term and dependency features.

features. As we can see, all methods (variant of kNN, weighted sum, SVM and LDA) have better recommendation results than the baseline algorithm. More specifically, we find that our proposed kNN variant beats other iterative methods, and the LDA based method has the best overall performance.

Table 6.6: Relationship between virtual rating generation effect and users' historical rating number

history size	Dataset					
	(1)	(2)	(3)	(4)	(5)	(6)
20	25.49	36.84	30.15	42.02	33.33	36.84
50	5.60	9.09	10.81	11.62	12.76	11.59
100	8.33	21.05	8.33	10.41	11.11	8.43
150	0.00	10.00	20.00	30.00	48.27	39.62

Meanwhile, according to the user-based MAE metric, we analyze the relationship between the user history rating size and the virtual rating effect. As in Table 6.6, we find that those users with history rating number less than 50 get more reduced user-based MAE values. The table shows the percentage of users who have a significant U-MAE reduce (reduced more than 30%). This shows that the primary contributions of virtual ratings goes toward users with lean rating histories, and our virtual rating generation algorithm is more meaningful for these users, in terms of getting lower error metric values.

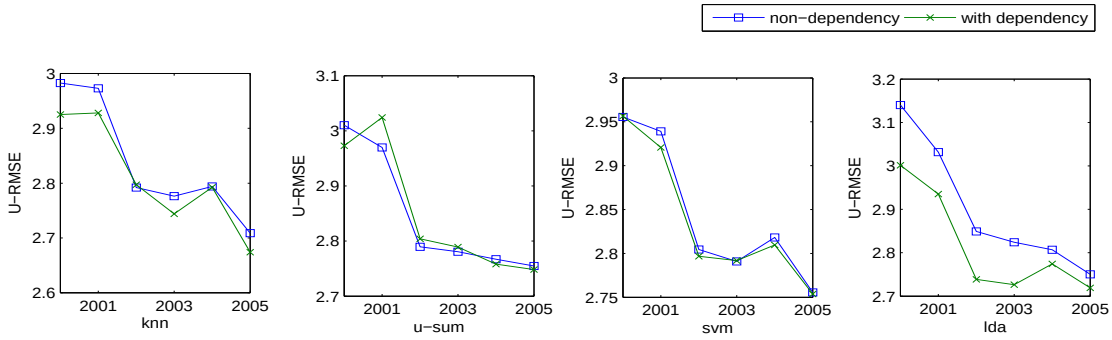


Figure 6.6: Comparison of the experimental results from single-term feature set (BOW) and the results from compounded feature set including dependency information.

6.3.5 Factor Analysis

Usage of dependencies: We also notice that by using different feature sets, the effect of virtual rating generating strategies also can be very different. We compared the experimental results by using single-term features only and using all features including dependencies. All the results show that using the features with dependency is an important factor for improving recommendation quality. In Figure 6.6, we show four method's performance on all six datasets, given different feature sets as input. U-RMSE is used as evaluation metric.

Usage of virtual ratings: Finally, we investigate the effect of virtual rating usage in both similarity matrix construction and rating prediction (see Section 6.2.3). We not only analyze the meaning of virtual ratings in calculating the movie-movie similarity, but also we can treat virtual rating as a user's real history rating data to predict future movie preference. As we can see in Table 6.7, when the virtual ratings are not involved in either the similarity construction, or the rating history, the result is the baseline ("M_o H_o"), and when they serve only as history ratings ("M_o H_v"), the accuracy does not improve too much. However, when the virtual ratings are used for computing the item similarity matrix, no matter whether the virtual ratings serve as history ratings or not, the accuracy is improved significantly ("M_v H_o", "M_v H_v"). We hence conclude that virtual ratings play an important role in our system, especially in improving the item similarity matrix computation. In our experiment, we use virtual ratings in both of them ("M_v H_v" in Table 6.7).

6.3.6 Supplemental Experiments

As we mentioned before, in the iterative strategy, although we believe that the set of feature terms becomes a constant collection when the dataset is large enough, we do test the performance when

Table 6.7: Virtual rating usage analysis for U-MAE results of kNN variant. (M: similarity matrix, H: history rating; o: use only original ratings; v: virtual ratings used)

Method	Dataset					
	(1)	(2)	(3)	(4)	(5)	(6)
M_o H_o	2.5637	2.6194	2.5004	2.4910	2.4886	2.3726
M_o H_v	2.5610	2.5732	2.4954	2.4396	2.4463	2.3576
M_v H_o	2.4617	2.4800	2.4325	2.4090	2.4552	2.3021
M_v H_v	2.4725	2.5073	2.4305	2.3638	2.3984	2.2953

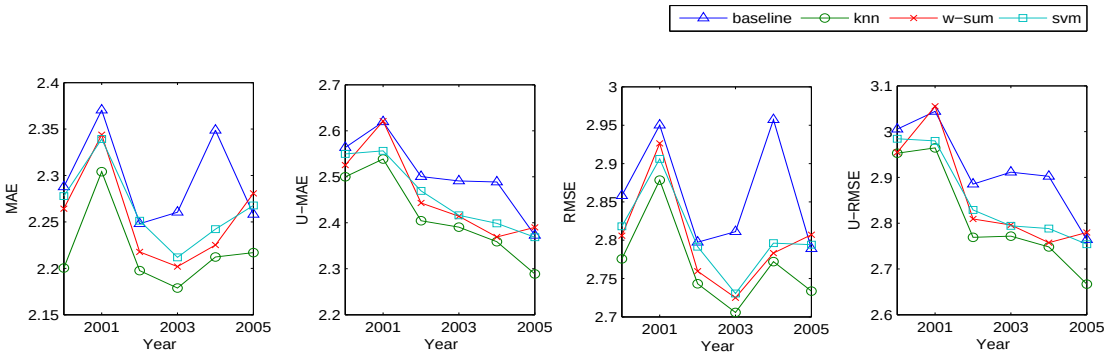


Figure 6.7: Comparison of results from three iterative virtual rating generating methods with the baseline result by using four different evaluation metrics. All methods use features generated from training dataset only, and the kNN variant method has the lowest error metric value and best performance.

all the features are generated from the training dataset (rated comments before the time boundary) only. As we can see in Figure 6.7, we use the same datasets as those used in the experiment above, the only difference is that all the features are from the training dataset only. Experimental results show that all three iterative methods beat the baseline, while the kNN variant method is still the best one among them. It proves that virtual ratings are a major contributor to the lowered error metric values.

6.4 Conclusions and Future Work

Our work has introduced a framework for improving conventional recommender system performance by utilizing external textual information. We use virtual ratings generated from unrated comments to help predict future ratings. However, this should not be limited in the sentiment review scenario. Scaling any topic-related text could be applied, since we used general topic mode

based methods in our framework. We tested three iterative virtual rating generating methods and one topic model based method. We also conducted analysis on feature selection quality. To have a better way to evaluate and analyze the results, we use user-based metrics in place of the traditional metrics. Our experimental results show that the kNN variant virtual rating generating method and the LDA-based method perform the best, and the discriminative dependencies from supervised collocation mining can be treated as an important linguistic feature for iterative methods.

Due to the relatively straightforward way in which we extract the single-term and dependency features, our methods can be easily applied to other domains. In addition, as stated earlier, the virtual ratings can be incorporated into many other recommendation algorithms. Finally, analyzing the unrated text from a global perspective is also worthy of study.

Chapter 7

Conclusions and Future Work

In this dissertation, we have discussed the functionality of dependency parses as textual features in text mining. Dependency parses can be used as textual features in many ways, either treated as general features, or used to construct a graph representation of the text, or used to extract the discriminative collocations.

When used as general features, dependency parses incorporate more linguistic information than the classic bag-of-words model, or n-gram models. For example, by using dependency parsing, we can find meaningful word-word combinations even though they are very far away from each other lexically in a sentence. Secondly, constructing the graph representation for the text through dependency parsing is helpful when the text is short, since keyword extraction through statistical methods is not feasible. Finally, we have proposed a supervised collocation mining algorithm, which extends classic collocation mining in natural language processing. By utilizing positive and negative training datasets, our algorithm can detect discriminative dependency parses as context information for classification or entity identification.

It should be noted that we have judiciously utilized the various representations in different applications: for instance, they are treated as general features such as in the movie spoiler case, or as graph features as in the text segmentation case, or as discriminative collocations as in the drug interaction case and in the movie recommender system example.

Application Scenarios: However, these three approaches are suitable for different scenarios.

1. In general, the first method, which simply augments dependency parses into the feature vector, can be applied to most situations. Each sentence is broken down into dependency parses with many syntactic relations, and these relations are then preprocessed by filtering out the trivial ones. We use these nontrivial syntactic relations just as the other terms. Enriching the feature vector by adding the relations can improve classification, clustering, and other traditional data mining contexts.

2. The second approach is suitable for short text analysis. This is because short text requires

extra features besides the term vector, and statistical methods usually do not perform well on short text. Parsing the sentences and using the syntactic information aids significantly in this respect. In the text segmentation application, we use the textual syntactic graph to calculate distances between sentences based on distances between sub-graphs. Furthermore, when a sentence or text snippet is represented as a graph, in addition to sub-graph distances, any graph-based algorithm can be employed for studying such graphs. For instance, we can utilize algorithms for organizing tag clouds that exploit the graph structure of sentences.

3. Finally, the supervised collocation is a more delicate case where dependency parses are used in a domain-specific context. In the drug interaction example, they are used to identify those sentences with drug interaction information, and in the movie recommender system case, they are helpful to classify the movie comments' sentiment polarity. Other domain-specific contexts are possible. For instance, dependency parses may be used to decide if social updates in online media fall into a certain topic area, such as whether an update on LinkedIn is a career-related one or something related to a person's personal life. By using supervised collocation mining, we may identify helpful linguistic cues to distinguish the professional ones from others, by identifying those that are connected exclusively with business terms for instance.

Limitations: It is noteworthy that since dependency parses are generated from parsing tools, this may limit our applications into certain areas. For example, off-line computation can handle large amounts of data, while online processing requires strict low latency, which remains a significant challenge to parsers. For those applications that require online classification, many existing parsers, such as the Stanford parser, may not be a good choice. Tradeoff between speed and accuracy has to be taken into account. Meanwhile, parsing Internet slang could be very difficult compared to well-formed language. Although symbols and abbreviations may not necessarily result in incorrect parsing, sentences with grammatical errors such as short tweets may confuse the parsers and yield unexpected parse trees. Nevertheless, our avoidance of more deeper semantic level features ensures that our methods, combined with ongoing improvement in parser tools and high-performance computing, should remain feasible and improve in popularity.

Future Improvement and Applications: As we mentioned above, dependency parse usage can be summarized into three categories: general features, graph features, and collocations. Significant new work can be done to both improve the ways in which we use dependency parses and in exploring new and interesting applications.

1. **Parser Customization (a)**: Current methods use different kinds of dependency parsers from the NLP arena, but the speed of these parsers is not competitive since completely constructing the parse from a sentence is a complicated task. However, dependency parses are not used as a whole since we filter out most of the trivial relations. In other words, we only care about a subset of syntactic relations among the original output, not the dependency parse tree itself. Based on the consideration above, if the dependency parsers can be customized so that only the non-trivial relations are generated, the performance of our methods could be

improved significantly. Thus, there is still room to improve the performance of the parsers to suit our applications.

2. **Parser Customization (b)**: As mentioned earlier, Internet slang and text riddled with abbreviations make the parsing task difficult. It will be very helpful if external knowledge of such 'netspeak' can be integrated into parsing tools.

3. **Weighting Improvement**: The current weighting schemes used in this dissertation are quite basic in nature. For the general use case where syntactic relations are added into the feature vectors, it is worth investigating the weighting difference between normal terms and syntactic virtual terms. Also, in our text segmentation example, we have proposed a basic weighting scheme for the text level graph based on dependency parses; other node weighting and edge weighting algorithms may be considered for specific domains.

4. **Visualization Integration**: As we mentioned in the text segmentation example, dependency parses conserve more information than BOW models. A text-level graph displays the connection strength between terms and also the layout among them in a natural way. Such information would be very useful if we consider integrating them into traditional visualizations. For instance, using term-term syntactic relations to improve layouts of tag clouds is an important future research problem.

5. **More Applications for Classification**: Besides the system improvement and the visualization application above, there are many text classification problems to which we can apply our supervised collocation approach, especially to detect specific types of text. One example is for professional news text detection. For example, social media, such as Twitter, Facebook, LinkedIn, generates large amounts of information every hour. If we desire to find valuable business-related news snippets out of many personal tweets talking about, say, dinner or travel, supervised collocation may be helpful (just as "when . . . used . . . with" helps us to identify drug interaction texts). Another example is for detecting 'sentiment' statements. In our recommender system example before, we used dependency parses to help generate virtual ratings. It also can be used directly to perform classification if pre-trained supervised collocations for certain type of sentiment text are available.

In summary, our work opens up many interesting avenues for further research. When serving as general textual features, dependency parses are currently chosen manually, whereas a deeper analysis for choosing dependency relations for different languages or different domains will be very helpful. A graph representation of text brings many ready-made solutions to text mining from the graph analysis arena. Not only can we do linear segmentation, but our approach also supports clustering and key phrase extraction. Finally, supervised collocation mining, as a special feature extraction method, could be applied to many other text mining problems than studied here. For example, using discriminative dependency parses as context information to do named entity recognition or word disambiguation is an interesting direction (e.g., Apple as a company may have different context dependency parses from the fruit apple). Another possible real life application

is to tackle the problem of characterizing news feeds in online media, tracking the genealogy of information, and in general raising the expressiveness with which we model and reason about text.

Bibliography

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[2] Russ B. Altman, Casey M. Bergman, Judith Blake, Christian Blaschke, Aaron Cohen, Frank Gannon, Les Grivell, Udo Hahn, William Hersh, Lynette Hirschman, Lars Jensen, Martin Krallinger, Barend Mons, Seán I. O'Donoghue, Manuel C. Peitsch, Dietrich Rebholz-Schuhmann, Hagit Shatkay, and Alfonso Valencia. Text mining for biology - the way forward: opinions from leading scientists. *Genome Biology*, 9(Suppl 2):S7+, 2008.

[3] Xavier Amatriain, Josep M. Pujol, Nava Tintarev, and Nuria Oliver. Rate it again: increasing recommendation accuracy by user re-rating. In Lawrence D. Bergman, Alexander Tuzhilin, Robin Burke, Alexander Felfernig, Lars S. Thieme, Lawrence D. Bergman, Alexander Tuzhilin, Robin Burke, Alexander Felfernig, and Lars S. Thieme, editors, *Proceedings of the 3rd ACM Conference on Recommender Systems*, pages 173–180. ACM, 2009.

[4] Jing Bai, Dawei Song, Peter Bruza, Jian-Yun Nie, and Guihong Cao. Query expansion using term relationships in language models for information retrieval. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 688–695, New York, NY, USA, 2005. ACM Press.

[5] Xinlong Bao, Lawrence Bergman, and Rich Thompson. Stacking recommendation engines with additional meta-features. In *Proceedings of the 3rd ACM Conference on Recommender Systems*, pages 109–116, New York, NY, USA, 2009. ACM.

[6] Justin Basilico and Thomas Hofmann. Unifying collaborative and content-based filtering. In *Proceedings of the 21st International Conference on Machine Learning*, page 9, New York, NY, USA, 2004. ACM.

[7] Doug Beeferman, Adam Berger, and John D. Lafferty. Statistical models for text segmentation. *Machine Learning*, 34(1-3):177–210, 1999.

[8] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, pages 585–591. MIT Press, 2001.

[9] Chaim M. Bell, Wendy V. Hatch, Hadas D. Fischer, Geta Cernat, J. Michael Paterson, Andrea Gruneir, Sudeep S. Gill, Susan E. Bronskill, Geoffrey M. Anderson, and Paula A. Rochon. Association Between Tamsulosin and Serious Ophthalmic Adverse Events in Older Men Following Cataract Surgery. *JAMA*, 301(19):1991–1996, May 2009.

[10] R. M. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Proceedings of the 7th IEEE International Conference on Data Mining*, pages 43–52, Washington, DC, USA, October 2007. IEEE Computer Society.

[11] Adam Berger and Harry Printz. Recognition performance of a large-scale dependency grammar language model. In *Int'l Conference on Spoken Language Processing (ICSLP'98*, pages 2083–2086, 1998.

[12] J. Bhogal, A. Macfarlane, and P. Smith. A review of ontology based query expansion. *Information Processing & Management*, 43(4):866–886, July 2007.

[13] Daniel M. Bikel. A distributional analysis of a lexicalized statistical parsing model. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 182–189, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[14] Bodo Billerbeck, Falk Scholer, Hugh E. Williams, and Justin Zobel. Query expansion using associated queries. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 2–9, New York, NY, USA, 2003. ACM.

[15] Chen Bin, Yang Xiaofeng, Su Jian, and Tan Chew Lim. Other-anaphora resolution in biomedical texts with automatically mined patterns. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 121–128, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[16] D. Blei, T. Gri, M. Jordan, and J. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *Proceedings of NIPS'04*, 2004.

[17] David M. Blei and John D. Lafferty. A correlated topic model of science. *Annals of Applied Statistics*, 1(1):17–35, 2007.

[18] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of machine learning research*, 3:993–1022, 2003.

[19] Jordan Boyd-Graber, Jonathan Chang, Sean Gerrish, Chong Wang, and David Blei. Reading tea leaves: How humans interpret topic models. In *Proceedings of NIPS'09*, 2009.

[20] Matthew Brand and Matthew Brand. Charting a manifold. In *Advances in Neural Information Processing Systems 15*, volume 15, pages 961–968, 2003.

[21] Curt Burgess, Kay Livesay, and Kevin Lund. Explorations in context space: Words, sentences, discourse. *Discourse Processes*, 25(2/3):211–257, 1998.

[22] Giuseppe Carenini and Rita Sharma. Exploring more realistic evaluation measures for collaborative filtering. In *Proceedings of the 19th National Conference on Artifical intelligence*, pages 749–754. AAAI Press / The MIT Press, 2004.

[23] Òscar Celma and Perfecto Herrera. A new approach to evaluating novel recommendations. In *Proceedings of the 2nd ACM Conference on Recommender Systems*, pages 179–186, New York, NY, USA, 2008. ACM.

[24] Yin-Wen Chang and Chih-Jen Lin. Feature ranking using linear SVM. In *JMLR: Workshop and Conference Proceedings*, pages 53–64.

[25] Ciprian Chelba, David Engle, Frederick Jelinek, Victor Jimenez, Sanjeev Khudanpur, Lidia Mangu, Harry Printz, Eric Ristad, Ronald Rosenfeld, Andreas Stolcke, and Dekai Wu. Structure and performance of a dependency language model. In *Proceedings of Eurospeech*, pages 2775–2778, 1997.

[26] Freddy Y. Y. Choi. Advances in domain independent linear text segmentation. In *Proceedings of NAACL-00*, 2000.

[27] Freddy Y. Y. Choi, Peter Wiemer-Hastings, and Johanna Moore. Latent semantic analysis for text segmentation. In *Proceedings of EMNLP*, pages 109–117, 2001.

[28] T. F. Cox, M. A. A. Cox, and B. Raton. Multidimensional scaling. *Technometrics*, 45(2):182, May 2003.

[29] Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines: Information Retrieval in Practice*. Addison Wesley, 1 edition, February 2009.

[30] Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma. Query expansion by mining user logs. *IEEE Transaction on Knowledge and Data Engineering*, 15:829–839, 2003.

[31] Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[32] Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, 1992.

[33] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 22(1):143–177, January 2004.

[34] Mona Diab and Philip Resnik. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 255–262, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[35] David L. Donoho and Carrie Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences of the United States of America*, 100(10):5591–5596, May 2003.

[36] Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Comput. Linguist.*, 19(1):61–74, March 1993.

[37] Stefan Evert. *Corpora and collocations*, volume A. 2008.

[38] Stefan Evert and Brigitte Krenn. Methods for the qualitative evaluation of lexical association measures. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 188–195, 2001.

[39] Olivier Ferret. Finding document topics for improving topic segmentation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 480–487, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[40] John R. Firth. A synopsis of linguistic theory, 1930-1955. *Studies in Linguistic Analysis*, pages 1–32, 1957.

[41] Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals Eugen.*, 7:179–188, 1936.

[42] Jonathan B. Fitzgerald, Birgit Schoeberl, Ulrik B. Nielsen, and Peter K. Sorger. Systems biology and combination therapy in the quest for clinical efficacy. *Nature Chemical Biology*, 2(9):458–466, August 2006.

[43] Gary W. Flake, Eric J. Glover, Steve Lawrence, and Lee C. Giles. Extracting query modifications from nonlinear SVMs. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 317–324, New York, NY, USA, 2002. ACM.

[44] Imola Fodor. A survey of dimension reduction techniques, 2002.

[45] Bruno M. Fonseca, Paulo Golgher, Bruno Pôssas, Berthier Ribeiro-Neto, and Nivio Ziviani. Concept-based interactive query expansion. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 696–703, New York, NY, USA, 2005. ACM.

[46] Katrin Fundel, Robert Kuffner, and Ralf Zimmer. RelEx–relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371, February 2007.

[47] Michel Galley, Kathleen McKeown, Eric Fosler-Lussier, and Hongyan Jing. Discourse segmentation of multi-party conversation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 562–569, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[48] Jianfeng Gao, Jian-Yun Nie, Guangyuan Wu, and Guihong Cao. Dependence language model for information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 170–177, New York, NY, USA, 2004. ACM.

[49] Niye Ge, John Hale, and Eugene Charniak. A statistical approach to anaphora resolution. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 161–171, 1998.

[50] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Comput. Linguist.*, 28(3):245–288, September 2002.

[51] Namrata Godbole, Manjunath Srinivasaiah, and Steven Skiena. Large-scale sentiment analysis for news and blogs. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, 2007.

[52] Richard L. Gorsuch. *Factor Analysis*. Lawrence Erlbaum, November 1983.

[53] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101 Suppl 1:5228–5235, April 2004.

[54] Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. Integrating topics and syntax. In *Proceedings of NIPS'05*, 2005.

[55] Thomas L. Griffiths, Mark Steyvers, and Joshua B. Tenenbaum. Topics in semantic representation. *Psychological Review*, 114(2):211–244, April 2007.

[56] Sheng Guo and Naren Ramakrishnan. Mining linguistic cues for query expansion: applications to drug interaction search. In *Proceeding of the 18th ACM Conference on Information and Knowledge Management*, pages 335–344, New York, NY, USA, 2009. ACM.

[57] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Mach. Learn.*, 46(1-3):389–422, 2002.

[58] Marti Hearst. Multi-paragraph segmentation of expository text. In *32nd. Annual Meeting of the Association for Computational Linguistics*, pages 9–16, New Mexico State University, Las Cruces, New Mexico, 1994.

[59] Marti A. Hearst. Texttiling: segmenting text into multi-paragraph subtopic passages. *Comput. Linguist.*, 23(1):33–64, March 1997.

[60] Marti A. Hearst. Untangling text data mining. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 3–10, Stroudsburg, PA, USA, 1999. Association for Computational Linguistics.

[61] Gregor Heinrich. Parameter estimation for text analysis. Technical report, University of Leipzig, 2008.

[62] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, January 2004.

[63] William R. Hersh, Ravi T. Bhupatiraju, and Susan Price. Phrases, boosting, and query expansion using external knowledge resources for genomic information retrieval. In *TREC*, pages 503–509, 2003.

[64] Heiko Hoffmann. Kernel PCA for novelty detection. *Pattern Recogn.*, 40:863–874, March 2007.

[65] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of UAI'99*, 1999.

[66] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 263–272, Washington, DC, USA, December 2008. IEEE Computer Society.

[67] Zan Huang, Daniel Zeng, and Hsinchun Chen. A comparison of collaborative-filtering recommendation algorithms for e-commerce. *IEEE Intelligent Systems*, 22(5):68–78, September 2007.

[68] Aminul Islam and Diana Inkpen. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Trans. Knowl. Discov. Data*, 2(2):1–25, July 2008.

[69] J. Edward Jackson. *A User's Guide to Principal Components*. Wiley Series in Probability and Statistics. Wiley-Interscience, September 2003.

[70] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, October 2002.

[71] Xiang Ji and Hongyuan Zha. Domain-independent text segmentation using anisotropic diffusion and dynamic programming. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 322–329, New York, NY, USA, 2003. ACM.

[72] Yufeng Jing and Croft. An association thesaurus for information retrieval. In *Proceedings of RIAO*, pages 146–160, Rockefeller University, New York, 1994.

[73] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[74] Richard Johansson and Pierre Nugues. Dependency-based semantic role labeling of propbank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 69–78, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[75] I. T. Jolliffe. *Principal Component Analysis*. Springer, New York, 2nd edition, October 2002.

[76] Sandra Kĺźbler, Ryan McDonald, and Joakim Nivre. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 2(1):1–127, 2009.

[77] Athanasios Kehagias, Fragkou Pavlina, and Vassilios Petridis. Linear text segmentation using a dynamic programming algorithm. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 1*, EACL '03, pages 171–178, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[78] Myoung-Cheol Kim and Key-Sun Choi. A comparison of collocation-based similarity measures in query expansion. *Information Processing & Management*, 35(1):19–30, January 1999.

[79] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, August 2009.

[80] John M. Kovarik, Doris Beyer, and Robert L. Schmouder. Everolimus drug interactions: application of a classification system for clinical decision making. *Biopharmaceutics & Drug Disposition*, 27(9):421–426, 2006.

[81] Hideki Kozima. Text segmentation based on similarity between words. In *Proceedings of the Association for Computational Linguistics*, pages 286–288, 1993.

[82] Deept Kumar, Naren Ramakrishnan, Richard F. Helm, and Malcolm Potts. Algorithms for storytelling. In *Proc. KDD'06*, pages 604–610, 2006.

[83] Stephane Lafon and Ann B. Lee. Diffusion maps and Coarse-Graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1393–1403, 2006.

[84] Man Lan, Chewlim Tan, and Hweeboon Low. Proposing a new term weighting scheme for text categorization. In *AAAI2006*, pages 763–768, 2006.

[85] Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse Processes*, pages 259–284, 1998.

[86] Ken Lang. Newsweeder: learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339. Morgan Kaufmann Publishers Inc.: San Mateo, CA, USA, 1995.

[87] Shalom Lappin and Herbert J. Leass. An algorithm for pronominal anaphora resolution. *Comput. Linguist.*, 20(4):535–561, December 1994.

[88] Martin H. C. Law and Anil K. Jain. Incremental nonlinear dimensionality reduction by manifold learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28:377–391, March 2006.

[89] Cane W. K. Leung, Stephen C. F. Chan, and Fu-Lai Chung. Evaluation of a rating inference approach to utilizing textual reviews for collaborative recommendation. In *Proceedings of the 4th International Conference (Cooperative Internet Computing)*, pages 94–109, Hong Kong, China, 2006. World Scientific.

[90] Dekang Lin. Principar - an efficient, broad-coverage, principle-based parser. In *COLING*, pages 482–488, 1994.

[91] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[92] B. Liu, Y. Dai, X. Li, W. Lee, and P. Yu. Building text classifiers using positive and unlabeled examples. In *Proceedings of ICDM'03*, 2003.

[93] Shuang Liu, Fang Liu, Clement Yu, and Weiyi Meng. An effective approach to document retrieval via utilizing wordnet and recognizing phrases. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 266–272, New York, NY, USA, 2004. ACM Press.

[94] Zhenyu Liu and Wesley W. Chu. Knowledge-based query expansion to support scenario-specific retrieval of medical free text. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1076–1083, New York, NY, USA, 2005. ACM.

[95] Yuanhua Lv and ChengXiang Zhai. Positional language models for information retrieval. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 299–306, New York, NY, USA, 2009. ACM.

[96] Igor Malioutov and Regina Barzilay. Minimum cut model for spoken lecture segmentation. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 25–32, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[97] Rila Mandala, Takenobu Tokunaga, and Hozumi Tanaka. Combining multiple evidence from different types of thesaurus for query expansion. In *Research and Development in Information Retrieval*, pages 191–197, 1999.

[98] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[99] Christopher D. Manning and Hinrich Schtze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, June 1999.

[100] Benjamin M. Marlin and Richard S. Zemel. Collaborative filtering and the missing at random assumption. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, 2007.

[101] Benjamin M. Marlin and Richard S. Zemel. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the 3rd ACM Conference on Recommender Systems*, pages 5–12, New York, NY, USA, 2009. ACM.

[102] M. Marneffe, B. Maccartney, and C. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC'06*, 2006.

[103] Elaine Marsh and Naomi Sager. Analysis and processing of compact text. In *Proceedings of the 9th conference on Computational linguistics - Volume 1*, COLING '82, pages 201–206, Czechoslovakia, 1982. Academia Praha.

[104] Irina Matveeva and Gina A. Levow. Topic segmentation with hybrid document indexing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 351–359, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[105] Bridget T. Mcinnes. Extending the log likelihood measure to improve collocation identification. Master's thesis, University of Minnesota, 2004.

[106] Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of the 16th International Conference on World Wide Web*, pages 171–180, New York, NY, USA, 2007. ACM.

[107] Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. Automatic labeling of multinomial topic models. In *Proceedings of KDD'07*, 2007.

[108] Prem Melville, Wojciech Gryc, and Richard D. Lawrence. Sentiment analysis of blogs by combining lexical knowledge with text classification. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1275–1284, New York, NY, USA, 2009. ACM.

[109] Rada Mihalcea and Courtney Corley. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of AAAI'06*, pages 775–780, 2006.

[110] Mandar Mitra, Amit Singhal, and Chris Buckley. Improving automatic query expansion. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 206–214, New York, NY, USA, 1998. ACM Press.

[111] Raymond J. Mooney and Razvan C. Bunescu. Mining knowledge from text using information extraction. *SIGKDD Explorations*, 7(1):3–10, June 2005.

[112] Jane Morris and Graeme Hirst. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Comput. Linguist.*, 17(1):21–48, March 1991.

[113] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, January 2007.

[114] Roberto Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):1–69, February 2009.

[115] Fernando A. Das Neves, Edward A. Fox, and Xiaoyan Yu. Connecting topics in document collections with stepping stones and pathways. In *CIKM*, pages 91–98, 2005.

[116] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pages 849–856. MIT Press, 2001.

[117] Masayuki Okabe and Seiji Yamada. Semisupervised query expansion with minimal feedback. *IEEE Trans. on Knowl. and Data Eng.*, 19(11):1585–1589, 2007.

[118] Satoshi Oyama, Takashi Kokubo, and Toru Ishida. Domain-specific web search with keyword spices. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):17–27, 2004.

[119] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.

[120] Denis Parra and Peter Brusilovsky. Collaborative filtering for social tagging systems: an experiment with CiteULike. In *Proceedings of the 3rd ACM conference on Recommender systems*, pages 237–240, New York, NY, USA, 2009. ACM.

[121] Helen J. Peat and Peter Willett. The limitations of term co-occurrence data for query expansion in document retrieval systems. *Journal of the American Society for Information Science*, 42(5):378–383, 1991.

[122] Pavel Pecina and Pavel Schlesinger. Combining association measures for collocation extraction. In *Proceedings of the COLING/ACL*, pages 651–658, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[123] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 433–440, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[124] Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April 2007. Association for Computational Linguistics.

[125] Lev Pevzner and Marti A. Hearst. A critique and improvement of an evaluation metric for text segmentation. *Comput. Linguist.*, 28(1):19–36, March 2002.

[126] István Pilászy and Domonkos Tikk. Recommending new movies: even a few ratings are more valuable than metadata. In *Proceedings of the 3rd ACM Conference on Recommender Systems*, pages 93–100, New York, NY, USA, 2009. ACM.

[127] S. C. Piscitelli and K. D. Gallicano. Interactions among drugs for HIV and opportunistic infections. *The New England journal of medicine*, 344(13):984–996, March 2001.

[128] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[129] Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. Expanding domain sentiment lexicon through double propagation. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence*, pages 1199–1204, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[130] Yonggang Qiu and Hans-Peter Frei. Concept based query expansion. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–169, New York, NY, USA, 1993. ACM Press.

[131] Chris Quirk, Arul Menezes, and Colin Cherry. Dependency treelet translation: syntactically informed phrasal smt. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 271–279, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[132] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A Metric for Distributions with Applications to Image Databases. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, Washington, DC, USA, 1998. IEEE Computer Society.

[133] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, V40(2):99–121, November 2000.

[134] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988.

[135] Jasmin Saric, Lars J. Jensen, Rossitza Ouzounova, Isabel Rojas, and Peer Bork. Extraction of regulatory gene/protein networks from medline. *Bioinformatics*, 22(6):645–650, March 2006.

[136] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International World Wide Web Conference*, pages 285–295, 2001.

[137] L. K. Saul, K. Q. Weinberger, J. H. Ham, F. Sha, and D. D. Lee. Spectral methods for dimensionality reduction. *Semisupervised Learning. MIT Press: Cambridge, MA*, 2006.

[138] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10:1299–1319, July 1998.

[139] Darren P. School, Darren Pearce, and Bn Qh. A comparative evaluation of collocation extraction techniques. In *Third International Conference on Language Resources and Evaluation, Las*, 2002.

[140] Hinrich Schütze. Automatic word sense discrimination. *Comput. Linguist.*, 24(1):97–123, March 1998.

[141] Charity D. Scripture and William D. Figg. Drug interactions in cancer therapy. *Nature Reviews Cancer*, 6(7):546–558.

[142] Violeta Seretan, Luka Nerima, and Eric Wehrli. Multi-word collocation extraction by syntactic composition of collocation bigrams. In Nicolas Nicolov, Kalina Bontcheva, Galia Angelova, and Ruslan Mitkov, editors, *Recent Advances in Natural Language Processing III: Selected Papers from RANLP 2003*, Current Issues in Linguistic Theory, pages 91–100. John Benjamins, Amsterdam/Philadelphia, 2004.

[143] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[144] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 2000.

[145] Frank Smadja. Retrieving collocations from text: Xtract. *Computational Linguistics*, 19:143–177, 1993.

[146] L. Smith, T. Rindflesch, and W. J. Wilbur. MedPost: a part-of-speech tagger for bioMedical text. *Bioinformatics*, 20(14):2320–2321, September 2004.

[147] Fei Song and W. Bruce Croft. A general language model for information retrieval. In *CIKM '99: Proceedings of the eighth international conference on Information and knowledge management*, pages 316–321, New York, NY, USA, 1999. ACM.

[148] Mark Steyvers and Tom Griffiths. *Probabilistic Topic Models*. Lawrence Erlbaum Associates, 2007.

[149] Mark Steyvers, Padhraic Smyth, Michal R. Zvi, and Thomas Griffiths. Probabilistic author-topic models for information discovery. In *Proceedings of KDD'04*, 2004.

[150] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:1–20, 2009.

[151] Masataka T. *Network analysis of adverse drug interactions*. Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji, Kyoto, Japan, 2008.

[152] Hirotoshi Taira and Masahiko Haruno. Feature selection in SVM text categorization. In *AAAI '99/IAAI '99*, pages 480–486, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[153] Teh, Yee Whye, Jordan, I. Michael, Beal, J. Matthew, Blei, and M. David. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, December 2006.

[154] Yee W. Teh, David Newman, and Max Welling. A collapsed variational bayesian inference algorithm for latent dirichlet allocation. In *Proceedings of NIPS'07*, 2007.

[155] Joshua B. Tenenbaum, Vin Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[156] Li Teng, Hongyu Li, Xuping Fu, Wenbin Chen, and I-Fan Shen. Dimension reduction of microarray data based on local tangent space alignment. In *Proceedings of the Fourth IEEE International Conference on Cognitive Informatics*, pages 154–159, Washington, DC, USA, 2005. IEEE Computer Society.

[157] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846, 1998.

[158] Akhmed Umyarov and Alexander Tuzhilin. Improving rating estimation in recommender systems using aggregation- and variance-based hierarchical models. In *Proceedings of the 3rd ACM Conference on Recommender Systems*, pages 37–44, New York, NY, USA, 2009. ACM.

[159] Masao Utiyama and Hitoshi Isahara. A statistical model for domain-independent text segmentation. In *Proceedings of the Association for Computational Linguistics*, pages 491–498, 2001.

[160] L. J. P. van der Maaten, E. O. Postma, and H. J. van den Herik. Dimensionality reduction: A comparative review. Technical Report TiCC-TR 2009-005, Tilburg University, 2009.

[161] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[162] Olga Vechtomova, Stephen Robertson, and Susan Jones. Query expansion with long-span collocates. *Information Retrieval*, 6(2):251–273, April 2003.

[163] H. M. Wallach. *Structured topic models for language*. PhD thesis, University of Cambridge, 2008.

[164] Hanna M. Wallach. Topic modeling: beyond bag-of-words. In *Proceedings of ICML'06*, 2006.

[165] Xiaojun Wan and Yuxin Peng. The earth mover's distance as a semantic measure for document similarity. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 301–302, New York, NY, USA, 2005. ACM.

[166] J. Wang, Z. Zhang, and H. Zha. Adaptive Manifold Learning. *Advances in Neural Information Processing Systems*, 2004.

[167] Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of the EMNLP-CoNLL*, pages 22–32, Prague, Czech Republic, 2007. Association for Computational Linguistics.

[168] Xuanhui Wang and Chengxiang Zhai. Mining term association patterns from search logs for effective query reformulation. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge mining*, pages 479–488, New York, NY, USA, 2008. ACM.

[169] Xuerui Wang and Andrew Mccallum. A note on topical n-grams. Technical Report UM-CS-2005-071, University of Massachusetts Amherst, 2005.

[170] Xuerui Wang, Andrew McCallum, and Xing Wei. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Proceedings of ICDM'07*, 2007.

[171] Xing Wei and Bruce W. Croft. LDA-based document models for ad-hoc retrieval. In *Proceedings of SIGIR'06*, 2006.

[172] Yair Weiss. Segmentation using eigenvectors: A unifying view. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 975–, Washington, DC, USA, 1999. IEEE Computer Society.

[173] Joachim Wermter and Udo Hahn. Collocation extraction based on modifiability statistics. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[174] Marc Wichterich, Ira Assent, Philipp Kranen, and Thomas Seidl. Efficient emd-based similarity search in multimedia databases via flexible dimensionality reduction. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 199–212, New York, NY, USA, 2008. ACM.

[175] Jinxi Xu and Bruce W. Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, 1996.

[176] Jinxi Xu and Bruce W. Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Systems*, 18:79–112, 2000.

[177] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49, New York, NY, USA, 1999. ACM Press.

[178] Zhenyue Zhang and Hongyuan Zha. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM J. Sci. Comput.*, 26:313–338, January 2005.