Aspect and Entity Extraction from Opinion Documents

BY

LEI ZHANG

B.S., Wuhan University, 2002
M.S., Wuhan University, 2005

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2012

Chicago, Illinois

Defense Committee:

Prof. Bing Liu, Chair and Advisor
Prof. Isabel Cruz
Prof. Clement Yu
Prof. Piotr Gmytrasiewicz
Dr. Chi Zhou Motorola Mobility Inc.

To my parents, my wife Liu and son Laurence

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my advisor Prof. Bing Liu for giving me the precious opportunity to pursue my doctoral studies under his guidance. It has been an honor to be his Ph.D. student. I appreciate all his contributions of time, ideas, funding and help to make my Ph.D. experience productive and stimulating.

I would also like to thank my other thesis committee members, Prof. Isabel Cruz, Prof. Clement Yu, Prof. Piotr Gmytrasiewicz and Dr. Chi Zhou, for their support and assistance.

I want to thank my classmates and colleagues in UIC for their friendship and support, Xiaowen Ding, Nitin Jindal, Arjun Mukherjee, Jia Chen, Wenxuan Gao, Wanzhi Zhang, Yan Xie, Lifeng Jia, Juzheng Li, Fei Dong, Li Lu and many other people. My study in UIC has been a wonderful experience.

LZ

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

LIST OF TABLES

LIST OF TABLES(CONTINUED)

LIST OF FIGURES

FIGURE PAGE

LIST OF ABBREVIATIONS

CRF	Conditional Random Fields
EM	Expectation-Maximization
HITS	Hyperlink-induced Topic Search
HMM	Hidden Markov Model
IE	Information Extraction
LDA	Latent Dirichlet Allocation
MRE	Mutual Reinforcement Based on Expected Values
NB	Naïve Bayes
NLP	Natural Language Processing
NER	Named Entity Recognition
PLSA	Probabilistic Latent Semantic Analysis
POS	Part of Speech
PU	Positive and Unlabeled

SUMMARY

Opinion mining or sentiment analysis is the computational study of people's opinions, appraisals, attitudes, and emotions toward entities and their aspects. The entities usually refer to individuals, events, topics, products and organizations. The aspects are attributes or components of the entities. Opinion mining has been an active research area in Web mining and Natural Language Processing (NLP) in recent years. With the explosive growth of opinion documents (i.e., reviews, blogs, forum discussion posts and tweets) on the Web, individuals and organizations are increasingly using the content from these media for their decision making.

In this thesis, we present a comprehensive study of the automatic extraction of aspects and entities from opinion documents, which are essential for opinion mining. At the beginning, we briefly introduce the aspect-based opinion mining model. Then, we propose a new unsupervised method for product aspect extraction and ranking. It extracts product aspects from opinion documents based on language patterns and dependency grammar. Meanwhile, it is capable of ranking extracted aspects by their importance, i.e. relevancy and frequency. In addition, we discover that there are two kinds of special product aspects in some domains. One is noun aspect implying opinion. The other is the resource term. Novel extraction algorithms are proposed to identify them from opinion documents.

In terms of entity extraction task, it is similar to the classic named entity extraction (NER) problem. However, there is a major difference. In a typical opinion mining application, the users often want to find opinions on some competing entities, e.g., competing or relevant products. However, they often can only provide a few names as there are too many of them. The opinion

mining system has to find the rest from a corpus. This implies that the discovered entities must be of the same type/class. Generally, this is a set expansion problem. To deal with this problem, we present two set expansion algorithms for entity extraction in opinion documents. One is based on *positive and unlabeled* (PU) learning model. The other is based on *Bayesian Sets*.

We also discuss extracting topic documents from a collection. Opinion mining system crawls and indexes opinion documents first, which are used for different specific tasks later. Typically, the documents are not well categorized because one does not know what the future tasks will be. Normally, when a user wants to study consumer opinions on a type of products, keyword search is used to find relevant opinion documents for analysis. However, the documents that are retrieved in this way can have both low recall and low precision. Another way is to train a document classifier. But the training procedure is time-consuming and labor-intensive, sometimes formidable. We propose an unsupervised technique to solve this problem based on a new PU learning model.

1. INTRODUCTION

Opinion mining or sentiment analysis is the computational study of people's opinions, appraisals, attitudes, and emotions toward entities and their aspects. The entities usually refer to individuals, issues, events, topics, products and organizations. The aspects are attributes or components of the entities. Opinion mining has been an active research area in Web mining and Natural Language Processing (NLP) in recent years. It is not only technically challenging but also practically very useful. With the explosive growth of opinion documents (i.e., reviews, blogs, forum discussion posts and tweets) on the Web, individuals and organizations are increasingly using the content in these media for their decision making. However, people have difficulty, owing to their mental and physical limitations, producing consistent results when the amount of such information to be processed is huge. Automated opinion mining systems are thus needed, as subjective biases and mental limitations can be overcome with an objective opinion analysis system.

Researchers have studied opinion mining at different granularity levels. *Sentiment classification* is perhaps the most widely studied topic (Pang et al., 2002). It classifies an opinionated document (e.g., a product review) as expressing a positive or negative opinion. The task is also commonly known as the *document-level sentiment classification* because it considers the whole document as the basic information unit, which assumes that the document is known to be opinionated. Likewise, the sentiment classification can be applied to individual sentences. However, each sentence cannot be assumed to be opinionated in this case. We need to first classify a sentence as opinionated or not opinioned, which is called *subjectivity classification*.

The resulting opinionated sentences are classified as expressing positive or negative opinions. It is called the *sentence-level sentiment classification* (Wiebe and Riloff, 2005; Wiebe et al., 2004; Wilson et al., 2005).

Although opinion mining at document level and sentence level is useful in many cases, it still leaves much to be desired. A positive evaluative text on a particular entity does not mean that the author has positive opinions on every aspect of the entity. Also, a negative evaluative text does not mean that the author dislikes everything about the object. For example, in a product review, the reviewer usually writes both positive and negative aspects of the product, although the general sentiment on the product could be positive or negative. To obtain more fine-grained opinion analysis, we need to delve into the aspect level. This idea leads to *Aspect-based Opinion Mining* (Hu and Liu, 2004), whose basic task is to extract and summarize people's opinions expressed on entities or aspects of the entities. In this model, three basic tasks are required.

1. Identifying and extracting entities in evaluative texts
2. Identifying and extracting aspects of the entities
3. Determining sentiment polarities on entities or the aspects of entities

For example, in the sentence "*I bought a Sony camera yesterday, and its picture quality is great*," the aspect-based opinion analysis system should identify that the author expresses a positive opinion on the picture quality attribute of the Sony camera. Here *picture quality* is the aspect and *Sony camera* is the entity. In this thesis, we would like to study the first two tasks: aspect extraction and entity extraction. They are core components for the aspect-based opinion mining system. Note that some researchers use term *feature* (Hu and Liu, 2004) or *facets* (Mei et al., 2007) to mean aspect in their contexts and use term *object* to mean entity. Besides, in some research literature, people do not distinguish aspects and entities. They use the general term

opinion targets to represents all the objects, which opinion can be expressed on (Qiu et al., 2011; Jakob and Gurevych, 2010).

1.1. Significance of the Study

Generally speaking, aspect extraction and entity extraction fall to the broad class of Information Extraction (IE) (Sarawagi, 2008), whose goal is to automatically extract structured information (e.g. the names of persons, organizations and locations) from unstructured sources.

However, traditional information extraction techniques are often developed for formal genre (e.g. news, scientific papers), which do not apply effectively to opinion mining or sentiment analysis. We aim to automatically extract fine-grained information from opinion documents. Their data sources are often ungrammatical and noisy, containing spelling errors (e.g. improper capitalization), abbreviations, slang and emoticons. Aspect extraction and entity extraction remain to be challenging problems for opinion mining or sentiment analysis. On the other hand, they are critically important, because without knowing aspects and entities in a corpus, the mined opinions have little use.

1.2. Contribution

- We propose an unsupervised extraction and ranking approach for general aspects, Compared with one state-of-art method (Qiu et al., 2011), it has better performance. Moreover, it can rank product aspects based on their importance, which can help users to identify important aspects effectively.

- We propose novel approaches to identify two kinds of special product aspects in opinion documents. One is noun aspect implying opinions. The other is the resource term.

To the best of our knowledge, these problems have not been studies before in the research literature. However, these aspects are very important for opinion mining. Without identifying such aspects, the recall of opinion mining suffers.

- We study the entity extraction problem. For opinion mining, the users are interested at finding competing entities from opinion documents. Thus, we regard the entity extracting task as a set expansion problem. Two set expansion algorithms based on *PU* learning and *Bayesian Sets* are proposed to tackle the problem.

- We also discuss about topic document extraction in a collection. It is a closely related problem to aspect extraction and entity extraction. Currently, in many opinion mining systems, opinion documents are crawled and indexed first, and used for different specific tasks later. Typically, the documents are not well categorized because one does not know what the future tasks will be. When a user wants to study consumer opinions on a type of products, he/she has to find relevant opinion documents first for analysis. It is an important step. Without identifying correct topic opinion documents, the subsequent aspect extraction, entity extraction or sentiment analysis tasks will not produce reliable results. Keyword search is the primary approach. However, the documents that are retrieved in this way can have both low recall and low precision. An alternative method is to do supervised learning or classification. However, manual labeling of training data for each task is labor-intensive and time-consuming. We propose a novel technique to solve this problem without the need of any manually labeled training data.

2. ASPECT-BASED OPINION MINING MODEL

In this section, we give a brief introduction to aspect-based opinion mining model, which provides the background about aspect extraction and entity extraction tasks.

2.1. Model Concepts

Opinions can be expressed on anything, e.g., a product, a service, an individual, an organization, an event, or a topic, by any person or organization. We use the term *entity* to denote the target object that has been evaluated. An entity can have a set of components (or parts) and a set of attributes. Each component may have its own sub-components and its set of attributes, and so on. Thus, an entity can be hierarchically decomposed based on the *part-of* relation. Formally, we have the following definition (Liu, 2010):

Definition (entity): An *entity e* is a product, service, person, event, organization, or topic. It is associated with a pair, $e(T, W)$, where T is a hierarchy of *components* (or *parts*), *sub-components*, and so on, and W is a set of *attributes* of e. Each component or sub-component also has its own set of attributes.

Example: A particular brand of cellular phone is an entity, e.g., *iPhone*. It has a set of components, e.g., *battery* and *screen*, and also a set of attributes, e.g., *voice quality*, *size*, and *weight*. The battery component also has its own set of attributes, e.g., *battery life*, and *battery size*.

Based on this definition, an entity can be represented as a tree or hierarchy. The root of the tree is the name of the entity. Each non-root node is a component or sub-component of the

entity. Each link is a *part-of* relation. Each node is associated with a set of attributes. An opinion can be expressed on any node and any attribute of the node.

Example: One can express an opinion on the cellular phone itself (the root node), e.g., "*I do not like iPhone*", or on any one of its attributes, e.g., "*The voice quality of iPhone is lousy*". Likewise, one can also express an opinion on any one of the phone's components or any attribute of the component.

In practice, it is often useful to simplify this definition due to two reasons: First, natural language processing (NLP) is difficult. To effectively study the text at an arbitrary level of detail as described in the definition is very hard. Second, for an ordinary user, it is too complex to use a hierarchical representation. Thus, we simplify and flatten the tree to two levels and use the term *aspects* to denote both components and attributes. In the simplified tree, the root level node is still the entity itself, while the second level nodes are the different aspects of the entity.

Definition (aspect and aspect expression): The *aspects* of an entity *e* are the components and attributes of *e*. An *aspect expression* is an actual word or phrase that has appeared in text indicating an aspect.

Example: In the cellular phone domain, an aspect could be named *voice quality*. There are many expressions that can indicate the aspect, e.g., "*sound*", "*voice*", and also "*voice quality*".

Aspect expressions are usually nouns and noun phrases, but can also be verbs, verb phrases, adjectives, and adverbs. We call aspect expressions in a sentence that are nouns and noun phrases *explicit aspect expressions*. For example, "*sound*" in "*The sound of this phone is clear*" is an explicit aspect expression. We call aspect expressions of the other types, *implicit aspect expressions*, as they often imply some aspects. For example, "*large*" is an implicit aspect expression in "*This phone is too large*". It implies the aspect *size*. Many implicit aspect

expressions are adjectives and adverbs, which also imply some specific aspects, e.g., *expensive* (price), and *reliably* (reliability). Implicit aspect expressions are not just adjectives and adverbs. They can be quite complex, e.g., "*This phone will not easily fit in pockets*". Here, "*fit in pockets*" indicates the aspect *size* (and/or *shape*). In this thesis, we focus on extracting *explicit aspect expressions*, since most of aspect expressions in opinion documents are explicitly expressed.

Like aspects, an entity also has a name and many expressions that indicate the entity. For example, the brand *Motorola* (entity name) can be expressed in several ways, e.g., "*Moto*", "*Mot*" and "*Motorola*".

Definition (entity expression): an *entity expression* is an actual word or phrase that has appeared in text indicating an entity.

Definition (opinion holder): The *holder* of an opinion is the person or organization that expresses the opinion.

For product reviews and blogs, opinion holders are usually the authors of the postings. Opinion holders are more important in news articles as they often explicitly state the person or organization that holds an opinion.

We now turn to opinions. There are two main types of opinions: *regular opinions* and *comparative opinions*. Regular opinions are often referred to simply as opinions in the research literature. A comparative opinion expresses a relation of similarities or differences between two or more entities, which is usually expressed using the comparative or superlative form of an adjective or adverb. The discussion below focuses only on regular opinions. For simplicity, the terms *regular opinion* and *opinion* are used interchangeably below.

Basically, an opinion is a positive or negative view, attitude, emotion or appraisal about an entity or an aspect of the entity from an opinion holder. Positive, negative and neutral are called *opinion orientations*. Other names for opinion orientation are *sentiment orientation*, *semantic orientation*, or *polarity*. In practice, neutral is often interpreted as no opinion. We are now ready to formally define an opinion.

Definition (opinion): An *opinion* (or *regular opinion*) is a quintuple,

$$(e_i, a_{ij}, oo_{ijkl}, h_k, t_l),$$

where e_i is the name of an entity, a_{ij} is an aspect of e_i, oo_{ijkl} is the orientation of the opinion about aspect a_{ij} of entity e_i, h_k is the opinion holder, and t_l is the time when the opinion is expressed by h_k. The opinion orientation oo_{ijkl} can be positive, negative or neutral, or be expressed with different strength/intensity levels.

We now put everything together to define a model of entity, a model of opinionated document, and the mining objective, which are collectively called the *aspect-based opinion mining*.

Model of entity: An entity e_i is represented by itself as a whole and a finite set of aspects, $A_i = \{a_{i1}, a_{i2}, \ldots, a_{in}\}$. The entity itself can be expressed with any one of a final set of entity expressions $OE_i = \{oe_{i1}, oe_{i2}, \ldots, oe_{is}\}$. Each aspect $a_{ij} \in A_i$ of the entity can be expressed by any one of a finite set of aspect expressions $AE_{ij} = \{ae_{ij1}, ae_{ij2}, \ldots, ae_{ijm}\}$.

Model of opinionated document: An opinionated document d contains opinions on a set of entities $\{e_1, e_2, \ldots, e_r\}$ from a set of opinion holders $\{h_1, h_2, \ldots, h_p\}$. The opinions on each entity e_i are expressed on the entity itself and a subset A_{id} of its aspects.

Objective of opinion mining**:** Given a collection of opinionated documents D, discover all opinion quintuples (e_i, a_{ij}, oo_{ijkl}, h_k, t_l) in D.

2.2. Aspect-Based Opinion Summary

Most opinion mining applications need to study opinions from a large number of opinion holders. One opinion from a single person is usually not sufficient for action. This indicates that some form of summary of opinions is desirable. A common form of summary is based on aspects, and is called *aspect-based opinion summary* (or *feature-based opinion summary*). Generally, the discovered information is stored in database tables. Then a whole suite of database and visualization tools can be applied to see the results in all kinds of ways to gain insights of the opinions in structured forms and displayed as bar charts and/or pie charts. For example, the aspect-based summary can be visualized using a chart graph and opinions on different product aspects can also be compared in visualization as Figure 1, which has been used in many shopping websites.

Researchers have also studied opinion summarization in the tradition fashion, e.g., producing a short *text summary*. Such a summary gives the reader a quick overview of what people think about a product or service. A weakness of such a text-based summary is that it is not quantitative but only qualitative, which is usually not suitable for analytical purposes in practice. For example, a traditional text summary may say "*Most people do not like this product*". However, a quantitative summary may say that 70% of the people do not like this product and 30% of them like it. In most opinion mining applications, the quantitative side is crucial just like in the traditional survey research. In survey research, aspect-based summaries displayed as bar charts or pie charts are commonly used because they give the user a concise, quantitative and visual view. The user thus can find useful knowledge from visualizations conveniently. Instead of generating a

text summary directly from input reviews, it is also possible to generate a text summary based on the mining results from charts.

Figure 1. Opinion summary on different product aspects

3. ASPECT EXTRACTION

3.1. Data Format

Existing research on aspect extraction (more precisely, aspect expression extraction) is mainly carried out in online reviews. We thus focus on product reviews here. There are two common review formats on the Web.

Format 1 – **Pros, Cons and the detailed review**: The reviewer is asked to describe some brief pros and cons separately and also write a detailed/full review.

Format 2 – **Free format**: The reviewer can write freely, i.e., no separation of pros and cons.

To extract aspects from pros and cons in reviews of Format 1 (not the detailed review, which is the same as that in Format 2), many information extraction (IE) techniques can be applied. An important observation about pros and cons is that they are usually very brief, consisting of short phrases or sentence segments. Each sentence segment typically contains only one aspect, and sentence segments are separated by commas, periods, semi-colons, hyphens, *&*, *and*, *but*, etc. This observation helps the extraction algorithm to perform more accurately. Since aspect extraction from pros and cons is relatively simple, we will not discuss it further.

In this thesis, we focus on the more general case, i.e., extracting aspects from reviews of Format 2, which usually consist of complete sentences.

3.2. Related Work

We would like to introduce several main aspect extraction methods proposed in recent years. For these methods, we can group them into three categories: language rule mining, sequence models and topic modeling.

3.2.1. Language Rule Mining

Hu and Liu (2004) first proposed a technique to extract product aspect based on association rule mining. The main idea is that consumers often use the same words when they comment on the same product aspects, and then frequent itemsets of nouns in reviews are likely to be product aspects while the infrequent ones are less likely to be product aspects. The basic steps of the algorithm are as follows.

Step 1: Find frequent nouns and noun phrases. Nouns and noun phrases (or groups) are identified by a POS tagger. Only the frequent ones are kept. The reason for using this approach is that when people comment on different aspects of a product, the vocabulary that they use usually converges. Thus, those nouns that are frequently talked about are usually genuine and important aspects.

Step 2: Find infrequent aspects by exploiting the relationships between aspects and opinion words. The above step can miss many genuine aspect expressions which are infrequent. This step tries to find some of them. The idea is as follows: The same opinion word can be used to describe or modify different aspects. Opinion words that modify frequent aspects can also modify infrequent aspects, and thus can be used to extract infrequent aspects. For example, "picture" has been found to be a frequent aspect, and we have the sentence, "*The pictures are absolutely amazing*." If we know that "amazing" is an opinion word, then "software" can also be

extracted as an aspect from the following sentence, "*The software is amazing*." because the two sentences follow the same dependency pattern and "software" in the sentence is also a noun.

The idea of using the modifying relationship of opinion words and aspects to extract aspects can be generalized to using dependency relation. Zhuang et al. (2006) employed the dependency relation to extract aspect-opinion pairs from movie reviews. After parsed by a dependency relation parser (e.g. MINIPAR[1]), words in a sentence are linked to each other by a certain dependency relation. Figure 2 shows the dependency grammar graph of an example sentence, "*This movie is not a masterpiece.*", where "*movie*" and "*masterpiece*" have been labeled as aspect and opinion respectively, a dependency relation template could be found as the sequence "NN - nsubj - VB - dobj - NN". NN and VB are POS tags. Zhuang et al. (2006) first identified reliable dependency relation templates from training data, and then used them to identify valid aspect-opinion pairs in test data.

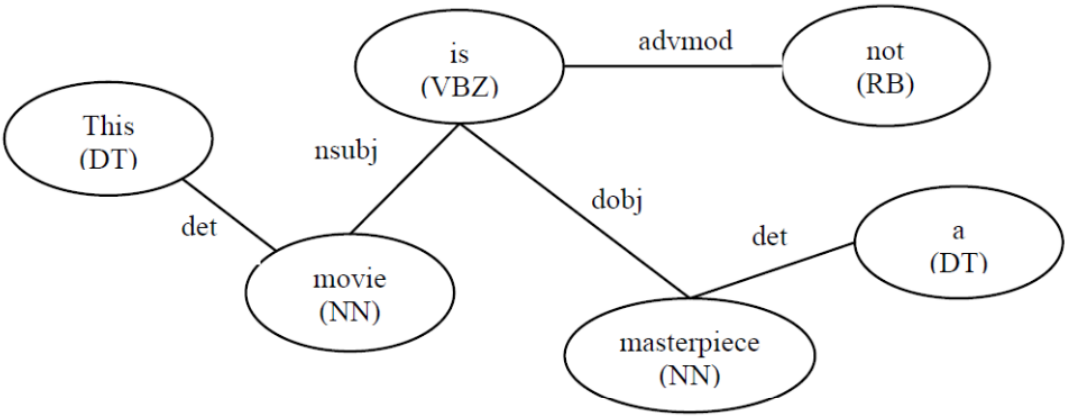

Figure 2. Dependency parsing for an example sentence

Double propagation (Qiu et al., 2011) further developed the idea. The method needs only an initial set of opinion word seeds as the input and no seed aspects are required. It is based on the observation that opinions almost always have targets, and there are natural relations

[1] http://webdocs.cs.ualberta.ca/~lindek/minipar.htm

connecting opinion words and targets in a sentence due to the fact that opinion words are used to modify targets. Furthermore, it was found that opinion words have relations among themselves and so do targets among themselves too. The opinion targets are usually aspects. Thus, opinion words can be recognized by identified aspects, and aspects can be identified by known opinion words. The extracted opinion words and aspects are utilized to identify new opinion words and new aspects, which are used again to extract more opinion words and aspects. This propagation or bootstrapping process ends when no more opinion words or aspects can be found. As the process involves propagation through both opinion words and aspects, the method is called *double propagation*. Extraction rules are designed based on different relations between opinion words and aspects, and also opinion words and aspects themselves. Dependency grammar was adopted to describe these relations. Double propagation works well for medium-size corpora. But for large and small corpora, it may result in low precision, and low recall respectively. To tackle such problems, we propose an unsupervised aspect extraction and ranking method, which will be discussed in detail in Section 3.3.

3.2.2. Sequence Models

Sequence models are widely used in information extraction tasks and can be applied to aspect extraction as well. Naturally, we can deem aspect extraction as a sequence labeling task, since product aspect, entity and opinion expression are often interdependent and occur at a sequence in a sentence.

Hidden Markov Model (HMM) is a directed sequence model for a wide range of time series data. It has been applied successfully for many sequence labeling problems such as named entity recognition (NER) in information extraction and part-of-speech (POS) tagging in NLP. In aspect extraction, we can regard words or phrases in review as observations and aspects or opinion expressions as underline states. Jin et al. (2009a and 2009b) utilized lexicalized HMM to

extract product aspects and opinion expressions from reviews. Different from traditional HMM, they integrate linguistic features such as part-of-speech and lexical patterns into HMM.

One limitation for HMM is that its assumptions may not adequately represent problems and lead to reduced performance. To address the limitation, Conditional Random fields (CRF) (Lafferty et al., 2001) is proposed. It is an undirected sequence model and can introduce more features than HMM at each time step. Jakob and Gurevych (2010) utilzied CRF to extract opinion targets from sentences which contain an opinion expression.

Similar work has been done in (Li et al., 2010). In order to model the long distance dependency with conjunctions (e.g. *and*, *or*, *but*) in sentences level and deep syntactic dependencies for aspects, positive opinions and negative opinions, they use skip-tree CRFs model to detect product aspects and opinions.

3.2.3. Topic Modeling And Clustering

Topic modeling methods have been attempted as an unsupervised and knowledge-lean approach. They exploit word occurrence information to capture latent topics in corpora. When applied, each discovered aspect is a unigram language model, i.e., a multinomial distribution over words. Such a representation is thus not as easy to interpret as aspects. But its advantage is that different words expressing the same or related aspects (more precisely aspect expressions) can usually be automatically grouped together under the same aspect. The topic models have been widely applied in text mining and natural language processing, for example, Probabilistic Latent Semantic Analysis (PLSA) and Latent Dirichlet Allocation (LDA).

Titov and McDonald (2008) pointed that global topic models such as PLSA and LDA might not be suitable for detecting rateable aspects. Both PLSA and LDA use the bag-of-words

representation of documents, therefore they can only explore co-occurrences at the document level. Thus, when the topic modeling methods are applied to a collection of reviews for different items, the extracted topics do not represent ratable aspects, but rather define clustering of the reviewed items into specific types/aspects. In order to tackle this problem, Titov and McDonald proposed multigrain topic models to discover local ratable aspects, which models two distinct types of topics: global topics and local topics. As in LDA, the distribution of global topics is fixed for a document (review). However, the distribution of local topics is allowed to vary across the documents. A word in the document is sampled either from the mixture of global topics or from the mixture of local topics or from the mixture of local topics specific to the local context of the word. It was demonstrated that ratable aspects will be captured by local topics and global topics will capture properties of reviewed items.

Lin and He (2009) proposed a joint topic-sentiment model, which extended LDA by adding a sentiment layer. It detects sentiment and aspect simultaneously from text. To separate aspects and opinion words using topic models, Brody and Elhadad (2010) proposed to first identify aspects using topic models and then identify aspect-specific opinion words by considering adjectives only. Zhao et al. (2010) proposed a MaxEnt-LDA hybrid model to jointly discover both aspect words and aspect-specific opinion words, which can leverage syntactic features to help separate aspects and opinion words. Jo and Oh (2011) proposed an *Aspect and Sentiment Unification* Model (ASUM) to model sentiments toward different aspects.

In (Su et al., 2008), the authors also proposed a clustering based method with mutual reinforcement to identify aspects. Similar work has been done in (Scaffidi et al., 2007), they proposed a language model approach to product aspect extraction with the assumption that product aspects are mentioned more often in a product review than they are mentioned in general English text. However, statistics may not be reliable when the corpus is small.

However, both topic modeling and clustering approaches are only able to find some general/rough aspects, and have difficulty in finding fine-grained or precise aspects.

3.3. Proposed Method

Our proposed method has two objectives. One is to mitigate the problems of *double propagation*, which does not perform well in large or small corpora. The other is to rank extracted aspects.

As introduced in Section 3.2.1, double propagation assumes that aspects are nouns/noun phrases and opinion words are adjectives. It propagates product aspects and opinion words at the same time. One of big advantages is that it requires no additional resources except an initial seed opinion lexicon, which is readily available (Wilson et al., 2005, Ding et al., 2008). Thus it is domain independent and unsupervised, avoiding laborious and time-consuming work of labeling data for supervised learning methods. It works very well in medium–size corpus. But for large corpora, this method may result in extracting many nouns/noun phrases which are not product aspects. The precision of the method plummets. The reason is that during propagation, adjectives which are not opinionated will be extracted as opinion words, e.g., "*entire*" and "*current*". These adjectives are not opinion words but they can modify many kinds of nouns/noun phrases, thus leading to extracting wrong aspects. Iteratively, more and more noises may be introduced during the process. The other problem is that for certain domains, some important aspects do not have opinion words modifying them. For example, in reviews of phone domain, a reviewer may say "*There is a camera on my phone*". Obviously, "*camera*" is an aspect, but the word "*camera*" may not be described by any opinion adjective, especially for a small corpus. Double propagation is not applicable under this situation.

To deal with such problems, we propose an approach, which consists of two steps: *aspect extraction* and *aspect ranking*. For aspect extraction, we still adopt the double propagation idea to populate aspect candidates. But two improvements based on *part-whole* relation patterns and "*no*" pattern are made to find aspects which double propagation cannot find. They can solve the recall problem partly. For aspect ranking, we rank aspect candidates by aspect importance.

Part-whole pattern indicates one object is part of another object. For the aforementioned example, we can find that it contains a part-whole relation between "*camera*" and "*phone*". "*camera*" belongs to "*phone*", which is indicated by the preposition "*on*". In this case, "*noun1 on noun2*" is a good pattern which implies noun1 is part of noun2. So if we know "*phone*" is the class concept, we can infer that "*camera*" is an aspect for "*phone*". There are many phrase or sentence patterns representing this semantic relation which was studied in (Girju et al., 2006). Beside part-whole patterns, "*no*" patterns is another important and specific aspect indicators in opinion documents. We will describe them in detail in the following section.

Given opinion word, *part-whole* pattern and "*no*" pattern, we have three strong aspect indicators at hands, but all of them are ambiguous, which means that they are not hard rules. We will inevitably extract wrong aspects (also called noises) by using them. Pruning noises from aspect candidates is a hard task. Instead, we propose a novel angle for solving this problem: aspect ranking. The basic idea is that we rank extracted aspect candidates by aspect importance. If an aspect candidate is correct and important, it should be ranked high. For unimportant aspect or noise, it should be ranked low in the final result. Basically, we transform the aspect extraction task from a classification problem to a ranking problem. Ranking is very useful in practice. In a large corpus, we may extract hundreds of fine-grained aspects. But the user often only cares about those important ones, which should be ranked high. We identified two major factors affecting the aspect importance: one is *aspect relevance* and the other is *aspect frequency*.

Aspect relevance: it describes how possible an aspect candidate is a correct aspect. We find that there are three strong clues to indicate aspect relevance in a corpus. The first clue is that a correct aspect is often modified by multiple opinion words (adjectives or adverbs). For example, in the mattress domain, "delivery" is modified by "quick" "cumbersome" and "timely". It shows that reviewers put emphasis on the word "delivery". Thus we can infer that "delivery" is a possible aspect. The second clue is that an aspect could be extracted by multiple part-whole patterns. For example, in the car domain, if we find following two phrases, "*the engine of the car*" and "*the car has a big engine*", we can infer that "*engine*" is an aspect for car, because both phrases contain part-whole relations to indicate "engine" is part of "car". The third clue is the combination of opinion word modification, part-whole pattern extraction and "no" pattern extraction. That is, if an aspect candidate is not only modified by opinion words but also extracted by part-whole or "no" patterns, we can infer that it is an aspect with high confidence. For example, for sentence "*there is a bad hole in the mattress*", it strongly indicates that "*hole*" is an aspect for a mattress because it is modified by opinion word "*bad*" and also in the part-whole pattern. What is more, we find that there is mutual enforcement relation between opinion word, part-whole pattern, "no" pattern and aspects. If an adjective modifies many correct aspects, it is highly possible to be a good opinion word. On the other hand, if an aspect candidate can be extracted by many opinion words, part-whole patterns, or "no" pattern, it is also highly possible to be a correct aspect. This indicates that the ranking algorithm HITS is applicable.

Aspect frequency: It is another important factor affecting aspect ranking (Blair-Goldensohn et al., 2008). We consider an aspect a_1 is more important than aspect a_2 if a_1 appears more frequently than a_2 in opinion documents. In practice, it is desirable to rank those frequent aspects higher than infrequent aspects. The reason is that missing a frequently mentioned aspect in opinion mining is very bad, but missing a rare aspect is not a big issue.

3.3.1. Part-Whole Pattern And "no" Pattern

As we discussed above, part-whole relation is a good indicator for aspect if class concept word (the "whole" part) is known. Let us denote the part-whole relation with PART(X, Y), where X is part of Y. For example, the compound nominal "*car hood*" contains the part-whole relation like PART (hood, car). If we know "car" is the class concept word, then we can infer that "*hood*" is an aspect for car. Part-whole patterns occur frequently in text and are expressed by a variety of lexico-syntactic structures (Girju et al, 2006). There are two types of lexico-syntactic structure to convey a part-whole relation: unambiguous structure and ambiguous structure. The unambiguous structure clearly indicates a part-whole relation. For example, for sentences "*the camera consists of lens, body and power cord.*" and "*the bed was made of wood*". In these cases, the simple detection of the patterns leads to the discovery of real part-whole relations. We can easily find aspects of the camera. Unfortunately, this kind of pattern is not frequent in our corpus. On the other hand, there are many ambiguous expressions that are explicit but convey part-whole relations only in some contexts. For example, for two phrases "*valley on the mattress*" and "*toy on the mattress*", "*valley*" is part of "*mattress*" whereas "*toy*" is not part of "*mattress*". Our idea is to use both the unambiguous and ambiguous patterns. Although ambiguous patterns may bring some noise, we can rank them low in the ranking procedure. For part-whole pattern, it includes following two sub patterns: phrase pattern and sentence pattern.

Phrase pattern: In this first case, the part and whole appear in the same phrase. We utilize the following patterns for opinion mining application.

(1) **NP + Prep + CP**: Noun phrase (NP) that contains the part noun and the class concept phrase (CP) that contains the whole noun are found in the same noun phrase. They are connected by the preposition word (Prep). For example, "*battery of the camera*" is an instance of this pattern where NP is the part noun (battery) and CP is the whole noun (camera). For our scenario, we only use three specific prepositions: "*of*", "*in*" and "*on*".

(2) **CP + with + NP**: Under this case, CP is the class concept word, NP is the noun phrase. They are connected by the word "*with*". Here NP is likely to be a feature. For example, in a phrase, "*mattress with a cover*", "*cover*" is the aspect for mattress.

(3) **NP CP or CP NP**: Noun phase (NP) and class phrase (CP) consist of a compound word. For example, "*mattress pad*". Here "pad" is the aspect of "*mattress*".

Sentence pattern: In this pattern, the part-whole relation is indicated in a sentence. The pattern contains specific verbs and the part and the whole can be found inside noun phrases or prepositional phrases which contain specific prepositions. We utilize the following pattern.

CP Verb NP: In this pattern, CP is the class concept phrase that contains the whole, NP is the noun phrase that contains the part and the verb is restricted and specific. For example, for sentence, "*the cars have doors*" is an instance of this pattern. We can infer that "*door*" is an aspect for "*car*". In sentence patterns, verbs play an important role. We use those verbs indicating part-of relations in a sentence, i.e., "have" "include" "contain" "consist", "comprise" and so on. For more details, please refer to paper (Girju et al, 2006). It is worth mention that in order to use part-whole relations, we have to find the class concept word for a corpus first. It is fairly easy because the noun with the most frequent occurrences in a domain is always the class concept word based on our experiments.

Besides opinion word and part-whole relation, "no" pattern is also an important pattern indicating aspect in a corpus. Here "no" represents word no. The basic form of the pattern is "no" word followed by noun/noun phrase. This simple pattern actually is very useful to aspect extraction. It is a specific pattern for Web reviews and forum posts. People often express their comments or opinions on aspects by this short pattern. For example, in a mattress domain, people always say that "*no noise*" and "*no indentation*". Here "*noise*" and "*indentation*" are all aspects for mattress. We discover that this pattern is frequently used in our corpora and a very good

indicator for aspects with a fairly high precision. But we have to take care of the some fixed "no" expression, like "*no problem*" "*no offense*". Under this situation, "*problem*" and "*offense*" should not be regarded as aspects. We have a list of such words, which are manually compiled.

3.3.2. Bipartite Graph And HITS Algorithm

Hyperlink-induced topic search (HITS) (Kleinberg, 1999) is a link analysis algorithm that rates Web pages. As discussed above, we can apply HITS algorithm to compute aspect relevance for ranking.

Before illustrating how HITS can apply to our scenario, let us give a brief introduction to HITS. Given a broad search query *q*, HITS sends the query to search engine system, then collects *k* (*k* = 200 in the original paper) highest ranked pages, which assume to be highly relevant to the search query. This set is called the root set *R*, then it grows *R* by including any page pointed to a page in *R*, then forms a base set *S*. HITS then works on the pages in *S*. It assigns every page in *S* an *authority score* and a *hub score*. Let the number of pages to be studied be *n*. We use $G = (V, E)$ to denote the (directed) link graph of *S*. *V* is the set of pages (or nodes) and *E* is the set of directed edges (or links). We use *L* to denote the adjacency matrix of the graph.

$$L_{ij} = \begin{cases} 1 & if\ (i,j) \in E \\ 0 & otherwise \end{cases} \quad (1)$$

Let the authority score of the page *i* be *A(i)*, and the hub score of page *i* be *H(i)*. The mutual reinforcing relationship of the two scores is represented as follows:

$$A(i) = \sum_{(j,i)\in E} H(j) \quad (2)$$

$$H(i) = \sum_{(i,j)\in E} A(j) \quad (3)$$

We can write them in the matrix form. We use A to denote the column vector with all the authority scores, A = $(A(1), A(2), \ldots, A(n))^T$, and use H to denote the column vector with all the hub scores, H = $(H(1), H(2), \ldots, H(n))^T$,

$$\mathrm{A} = L^T \mathrm{H} \quad (4)$$

$$\mathrm{H} = L\mathrm{A} \quad (5)$$

To solve the problem, the widely used method is power iteration, which starts with some random values for the vectors, e.g., $A_0 = H_0 = (1, 1, 1, \ldots 1,)$. It then continues to compute iteratively until the algorithm converges. From the formula, we can see that the authority score estimates the importance of the content of the page, and the hub score estimates the value of its links to other pages. An authority score is computed as the sum of the scaled hub scores that point to that page. A hub score is the sum of the scaled authority scores of the pages it points to. The key idea of HITS is that a good hub points to many good authorities and a good authority is pointed by many good hubs. Thus, authorities and hubs have a mutual reinforcement relationship.

For our scenario, we have three strong clues for aspects in a corpus: opinion words, part-whole patterns, and "no" pattern. Although all these three clues are not hard rules, there exist mutual enforcement relations between them. If an adjective modify many aspects, it is highly likely to be a good opinion word. On the other hand, if an aspect candidate is modified by many opinion words. It is highly likely to be a real aspect. The same goes with part-whole patterns, "no" pattern or the combination for these three clues. This kind of mutual enforcement relation can be naturally modeled in the HITS framework.

Naturally, aspect act as authorities and aspect indicators act as hubs. Different from the normal HITS algorithm, aspects only have authority scores and aspect indicators only have hub

scores in our case. They form a directed bipartite graph. We run the HITS algorithm on this bipartite graph. The basic idea is that if an aspect candidate has high-ranked authority score, it must be a highly-relevant aspect. If an aspect indicator has high-ranked hub score, it must be a good aspect indicator.

3.3.3. Aspect Ranking

Although HITS algorithm can rank aspects by aspect reference, the final ranking is not only determined by relevance. As we discuss before, aspect frequency is another important factor affecting the final ranking. It is highly desirable to rank those correct and frequent aspects at top because they are more important than the infrequent ones in opinion mining (or even other applications). With this in mind, we put everything together to present the final algorithm that we use. We use two steps:

Step 1: Compute aspect score without considering frequency using HITS. Initially, we use three aspect indicators to populate aspect candidates, which form a directed bipartite graph. Each aspect candidate acts as an authority node in the graph; each aspect indicator act as a hub node. For node *s* in the graph, we let H_s be the hub score and A_s be the authority score. Then, we initialize H_s and A_s to 1 for all nodes in the graph. We update the scores of H_s and A_s until they converge. Finally, we normalize A_s and compute the score *S* for an aspect.

Step 2: The final score considering the aspect frequency is given in Equation (6).

$$S = S(a)\log(freg(a)) \qquad (6)$$

where $freq(a)$ is the frequency count of aspect a, and $S(a)$ is the authority score of the candidate aspect *f*. The idea is to push the frequent candidate aspects up by multiplying the log of frequency. Log is taken in order to reduce the effect of big frequency count numbers.

3.3.4. Experiments

This section evaluates the proposed method. We first describe the data sets, evaluation metrics and then the experimental results. We also compare it with the *double propagation* method.

3.3.4.1. Data Sets

We used four diverse data sets to evaluate our techniques. They were obtained from a commercial company that provided opinion mining services to its industrial clients. Table 1 shows the domains (based on their names) and the number of sentences in each data set. The data in "Cars" and "Mattress" are product reviews extracted from some online review sites. "Phone" and "LCD" are forum discussions extracted from some online forum sites. We split each post into sentences and the sentences are POS-tagged using the Brill's tagger (Brill, 1995). The tagged sentences are the input to our system.

Data Sets	Cars	Mattress	Phone	LCD
# of Sentence	2223	13233	15168	1783

Table 1. Experimental data sets

3.3.4.2. Evaluation Metrics

Besides precision and recall, we adopt the precision@N metric (Pantel et al., 2009). It presents the percentage of correct aspects that are among the top N aspect candidates in a ranked list. We compare our method's results with those of double propagation which ranks extracted candidates only by occurrence frequency. We do not compare with other methods as it was shown in (Qiu et al., 2011) that double propagation performs better.

3.3.4.3. Experiment Results

We first compare our results with double propagation on recall and precision for different corpus sizes. The results are presented in Tables 2, 3, and 4 for the four data sets. They show the precision and recall of 1000, 2000, and 3000 sentences from these data sets. We did not try more sentences because manually checking the recall and precision becomes prohibitive. Note that there are less than 3000 sentences for "Cars" and "LCD" data sets. Thus, the columns for "Cars" and "LCD" are empty in Table 4. In the Tables, "DP" represents the double propagation method; "Ours" represents our proposed method)

	Cars		**Mattress**		**Phone**		**LCD**	
	Precision	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
DP	0.79	0.55	0.79	0.54	0.69	0.23	0.68	0.43
Ours	0.78	0.56	0.77	0.64	0.68	0.44	0.66	0.55

Table 2. Results of 1000 sentences

	Cars		**Mattress**		**Phone**		**LCD**	
	Precision	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
DP	0.70	0.65	0.70	0.58	0.67	0.42	0.64	0.52
Ours	0.66	0.69	0.70	0.66	0.70	0.50	0.62	0.56

Table 3. Results of 2000 sentences

	Cars	**Mattress**		**Phone**		**LCD**
		Precision	**Recall**	**Precision**	**Recall**	
DP		0.65	0.59	0.64	0.48	
Ours		0.66	0.67	0.62	0.51	

Table 4. Results of 3000 sentences

From the tables, we can see that for corpora in all domains, our method's recalls outperform double propagation with only a small loss in precision. In data sets for "Phone" and "Mattress", the precisions are even better. We also find that with increase of the data size, the recall gap between the two methods becomes smaller gradually and the precisions of both methods also drop. However, in this case, aspect ranking plays an important role in discovering important aspects through ranking.

Ranking comparison between the two methods is shown in Tables 5, 6, and 7, which give the precisions of top 50, 100 and 200 results respectively. Note that the experiments reported in these tables were run on the whole data sets. There were no more results for the "LCD" data beyond top 200 as there are only a limited number of aspects for "LCD" discussed in the data. So the column for the "LCD" in Table 7 is empty. We rank the extracted aspect candidates based on frequency for the double propagation method. As we discussed before, using occurrence frequency is the natural way to rank. The more frequent an aspect occurs in a corpus, the more important it is likely to be. But frequency-based ranking assumes the extracted candidates are correct features. The tables show that our proposed method outperforms double propagation considerably. The reason is that some highly-frequent aspect candidates extracted by double propagation are not correct aspects. Our method considers the aspect relevance as an important factor. So it produces much better rankings.

	Cars	**Mattress**	**Phone**	**LCD**
DP	0.84	0.81	0.64	0.68
Ours	0.94	0.90	0.76	0.76

Table 5. Precision at top 50

	Cars	**Mattress**	**Phone**	**LCD**
DP	0.82	0.80	0.65	0.68
Ours	0.88	0.85	0.75	0.73

Table 6. Precision at top 100

	Cars	**Mattress**	**Phone**	**LCD**
DP	0.75	0.71	0.70	
Ours	0.80	0.79	0.76	

Table 7. Precision at top 200

3.4. Identifying Noun Aspect Implying Opinion

We discover that in some product domains nouns and noun phrases that indicate aspects may also imply opinions. In many such cases, these nouns are not subjective but objective. Their involved sentences are also objective sentences and imply positive or negative opinions. To make

this concrete, let us see an example from a mattress review: "*Within a month, a valley formed in the middle of the mattress*." Here "*valley*" indicates the quality of the mattress (an aspect) and also implies a negative opinion. Identifying such nouns and noun phrases and their polarities is very challenging but critical for effective opinion mining in these domains. To the best of our knowledge, this problem has not been studied in the literature.

3.4.1. Related Work

Obviously, noun aspect implying opinion is a special kind of aspect. It is also closely related with opinion words, which are words that convey positive or negative polarities. They are critical for opinion mining (Pang et al., 2002; Hu and Liu, 2004; Wilson et al., 2005; Popescu and Etzioni, 2005; Ding et al., 2008; Titov and McDonald, 2008). The key difficulty in finding such words is that opinions expressed by many of them are domain or context dependent.

Several researchers have studied the problem of finding opinion words (Liu, 2010). The approaches can be grouped into corpus-based approaches (Hatzivassiloglou and McKeown, 1997; Qiu et al., 2011) and dictionary-based approaches (Hu and Liu 2004; Esuli and Sebastiani, 2006; Dragut et al., 2010). Dictionary-based approaches are generally not suitable for finding domain specific opinion words as dictionaries contain little domain specific information.

Hatzivassiloglou and McKeown (1997) did the first work to tackle the problem for adjectives using a corpus. The approach exploits some conjunctive patterns, involving *and*, *or*, *but*, *either-or*, or *neither-nor*, with the intuition that the conjoining adjectives subject to linguistic constraints on the orientation or polarity of the adjectives involved. Using these constraints, one can infer opinion polarities of unknown adjectives based on the known ones. Ding et al. (2008) introduced the concept of aspect context because the polarities of many opinion bearing words

sentence context dependent rather than just domain dependent. Qiu et al. (2011) proposed a method called *double propagation* that uses dependency relations to extract both opinion words and product aspects. However, none of these approaches handle nouns or noun phrases. Although Zagibalov and Carroll (2008) noticed the issue, they did not study it. Esuli and Sebastiani (2006) used WordNet to determine polarities of words, which can include nouns. However, dictionaries do not contain domain specific information.

3.4.2. Proposed Method

We start with some observations. For a product aspect with an implied opinion, there is either no adjective opinion word that modifies it directly or the opinion word that modify it usually have the same opinion.

Example 1: No opinion adjective word modifies the opinionated product aspect ("valley"): *"Within a month, a valley formed in the middle of the mattress."*

Example 2: An opinion adjective modifies the opinionated product aspect: *"Within a month, a bad valley formed in the middle of the mattress."*

Here, the adjective "*bad*" modifies "*valley*". It is unlikely that a positive opinion word will modify "*valley*", e.g., "*good valley*" in this context. Thus, if a product aspect is modified by both positive and negative opinion adjectives, it is unlikely to be an opinionated product aspect.

Based on these examples, we designed the following two steps to identify noun product aspects which imply positive or negative opinions:

Candidate Identification: This step determines the surrounding sentiment context of each noun aspect. The intuition is that if an aspect occurs in negative (respectively positive) opinion contexts significantly more frequently than in positive (or negative) opinion contexts, we can infer that its polarity is negative (or positive). A statistical test is used to test the significance. This step thus produces a list of candidate aspects with positive opinions and a list of candidate aspects with negative opinions.

Pruning: This step prunes the two lists. The idea is that when a noun product aspect is directly modified by both positive and negative opinion words, it is unlikely to be an opinionated product aspect.

Basically, step 1 needs the aspect-based sentiment analysis capability. We adopt the lexicon-based approach in (Ding et al. 2008) in this work.

3.4.2.1.Aspect-Based Sentiment Analysis

To use the lexicon-based sentiment analysis method, we need a list of opinion words, i.e., an opinion lexicon. Opinion words are words that express positive or negative sentiments. As noted earlier, there are also many words whose polarities depend on the contexts in which they appear. Researchers have compiled sets of opinion words for adjectives, adverbs, verbs and nouns respectively, called the *opinion lexicon*. In this thesis, we used the opinion lexicon complied by Ding et al. (2008). It is worth mentioning that our task is to find nouns which imply opinions in a specific domain, and such nouns do not appear in any general opinion lexicon.

Aggregating Opinions on an Aspect: Using the opinion lexicon, we can identify opinion polarity expressed on each product aspect in a sentence. The lexicon based method in (Ding et al. 2008) basically combines opinion words in the sentence to assign a sentiment to each product aspect. The sketch of the algorithm is as follows.

Given a sentence *s* which contains a product aspect *f* , opinion words in the sentence are first identified by matching with the words in the opinion lexicon. It then computes an orientation score for *f*. A positive word is assigned the semantic orientation (polarity) score of +1, and a negative word is assigned the semantic orientation score of -1. All the scores are then summed up using the following score formula:

$$score(f) = \sum_{w_i : w_i \in s \wedge w_i \in L} \frac{w_i.SO}{dis(w_i, f)} \qquad (7)$$

where w_i is an opinion word, L is the set of all opinion words (including idioms) and s is the sentence that contains the aspect *f*, and $dis(w_i, f)$ is the distance between aspect *f* and opinion word w_i in s. $w_i.SO$ is the semantic orientation (polarity) of word w_i. The multiplicative inverse in the formula is used to give low weights to opinion words that are far away from the aspect *f*.

If the final score is positive, then the opinion on the aspect in *s* is positive. If the score is negative, then the opinion on the aspect in *s* is negative.

Rules of Opinions: several language constructs need special handling, for which a set of rules is applied (Liu, 2010). A rule of opinion is an implication with an expression on the left and an implied opinion on the right. The expression is a conceptual one as it represents a concept, which can be expressed in many ways in a sentence.

Negation rule: A negation word or phrase usually reverses the opinion expressed in a sentence. Negation words include “no,” “not”, etc.

In this work, we also discovered that when applying negation rules, a special case needs extra care. For example, “*I am not bothered by the hump on the mattress*” is a sentence from a mattress review. It expresses a neutral feeling from the person. However, it also implies a negative opinion about “*hump*,” which indicates a product aspect. We call this kind of sentences

negated feeling response sentences. A sentence like this normally expresses the feeling of a person or a group of persons towards some items which generally have positive or negative connotations in the sentence context or the application domain. Such a sentence usually consists of four components: a noun representing a person or a group of persons (which includes personal pronoun and proper noun), a negation word, a feeling verb, and a stimulus word. Feeling verbs include "*bother*," "*disturb*," "*annoy*," etc. The stimulus word, which stimulates the feeling, also indicates an aspect. In analyzing such a sentence, for our purpose, the negation is not applied. Instead, we regard the sentence bearing the same opinion about the stimulus word as the opinion of the feeling verb. These opinion contexts will help the statistical test later.

*But clause rule***:** A sentence containing "but" also needs special treatment. The opinion before "but" and after "but" are usually the opposite to each other. Phrases such as "except that" and "except for" behave similarly.

Decreasing and increasing rules: These rules say that decreasing or increasing of some quantities associated with opinionated items may change the orientations of the opinions. For example, "*The drug eased my pain*". Here "*pain*" is a negative opinion word in the opinion lexicon, and the reduction of "*pain*" indicates a desirable effect of the drug. We have compiled a list of such words, which include "decrease", "diminish", "prevent", etc. The basic rules are as follows:

Decreased Neg → Positive
e.g: "*My problem have certainly diminished*"

Decreased Pos → Negative
e.g: "*These tires reduce the fun of driving.*"

Neg and Pos represent respectively a negative and a positive opinion word. Increasing rules do not change opinion directions.

Handing Context-dependent Opinions: As mentioned earlier, context-dependent opinion words (only adjectives and adverbs) must be determined by its contexts. We solve this problem by using the global information rather than only the local information in the current sentence. We use a conjunction rule. For example, if someone writes a sentence like "*This camera is very nice and has a long battery life*", we can infer that "long" is positive for "battery life" because it is conjoined with the positive word "nice". This discovery can be used anywhere in the corpus.

3.4.2.2. Determining Candidate Noun Aspects that Imply Opinions

Using the sentiment analysis method in above section, we can identify opinion sentences for each product aspect in context, which contains both positive-opinionated sentences and negative-opinionated sentences. We then determine candidate product aspects implying opinions by checking the percentage of either positive-opinionated sentences or negative-opinionated sentences among all opinionated sentences. Through experiments, we make an empirical assumption that if either the positive-opinionated sentence percentage or the negative-opinionated sentence percentage is significantly greater than 70%, we regard this noun aspect as a noun aspect implying an opinion. The basic heuristic for our idea is that if a noun aspect is more likely to occur in positive (or negative) opinion contexts (sentences), it is more likely to be an opinionated noun aspect. We use a statistic method *test for population proportion* to perform the significant test. The details are as follows. We compute the *Z*-score statistic with one-tailed test.

$$Z = \frac{p - p_0}{\sqrt{\frac{p_0(1 - p_0)}{n}}} \qquad (8)$$

where p_0 is the hypothesized value (0.7 in our case), p is the sample proportion, i.e., the percentage of positive (or negative) opinions in our case, and n is the sample size, which is the

total number of opinionated sentences that contain the noun aspect. We set the statistical confidence level to 0.95, whose corresponding *Z* score is -1.64. It means that *Z* score for an opinionated aspect must be no less than -1.64. Otherwise we do not regard it as an aspect implying opinion.

3.4.2.3. Pruning Non-Opinionated Aspects

Many of candidate noun aspects with opinions may not indicate any opinion. Then, we need to distinguish aspects which have implied opinions and normal aspects which have no opinions, e.g., "*voice quality*" and "*battery life*." For normal aspects, people often can have different opinions. For example, for "*voice quality*", people can say "*good voice quality*" or "*bad voice quality.*" However, for aspects with context dependent opinions, people often have a fixed opinion, either positive or negative but not both. With this observation in mind, we can detect aspects with no opinion by finding direct modification relations using a dependency parser. To be safe, we use only two types of direct relations:

Type1: *O* → *O-Dep* → *F*

It means *O* depends on *F* through a relation *O-Dep*. e.g: "*This TV has a good picture quality*."

Type 2: *O* → *O-Dep* → *H* ← *F-Dep* ← *F*

It means both *O* and *F* depends on *H* through relation *O-Dep and F-Dep* respectively. e.g: "*The springs of the mattress are bad*."

Here *O* is an opinion word, *O-Dep / F-Dep* is a dependency relation, which describes a relation between words, and includes *mod*, *pnmod, subj*, *s*, *obj*, *obj2* and *desc* (detailed explanations can be found in http://www.cs.ualberta.ca/~lindek/minipar.htm). *F* is a noun aspect. *H* means any word. For the first example, given aspect "*picture quality*", we can extract its modification opinion word "*good*". For the second example, given aspect "*springs*", we can get opinion word "*bad*". Here *H* is the word "*are*".

Among these extracted opinion words for the aspect noun, if some belong to the positive opinion lexicon and some belong to the negative opinion lexicon, we conclude the noun aspect is not an opinionated aspect and is thus pruned.

3.4.3. Experiments

We conducted experiments using four diverse real-life datasets of reviews. Table 8 shows the domains (based on their names) of the datasets, the number of sentences, and the number of noun aspects. The first two datasets were obtained from a commercial company that provides opinion mining services, and the other two were crawled by us.

Product Name	Mattress	Drug	Router	Radio
# Sentences	13191	1541	4308	2306
# Noun aspects	326	38	173	222

Table 8. Experimental datasets

An issue for judging noun aspects implying opinions is that it can be subjective. So for the gold standard, a consensus has to be reached between the two annotators.

For comparison, we also implemented a baseline method, which decides a noun aspect's polarity only by its modifying opinion words (adjectives). If its corresponding adjective is positive-orientated, then the noun aspect is positive-orientated. The same goes for a negative-orientated noun aspect. Then using the same techniques in Section 3.4.2.2 for statistical test (in this case, n in Equation (8) is the total number of sentences containing the noun aspect) and for pruning, we can determine noun aspects implying opinions from the data corpus.

Table 9 gives the experimental results. The performances are measured using the standard evaluation measures of precision and recall. From Table 9, we can see that the proposed method is much better than the baseline method on both the recall and precision. It indicates

many noun aspects that imply opinions are not directly modified by adjective opinion words. We have to determine their polarities based on contexts.

Product Name	Baseline		Proposed Method	
	Precision	Recall	Precision	Recall
Mattress	0.35	0.07	0.48	0.82
Drug	0.40	0.15	0.58	0.88
Router	0.20	0.45	0.42	0.67
Radio	0.18	0.50	0.31	0.83

Table 9. Experimental results for noun aspects

Product Name	Precision	Recall
Mattress	0.42	0.95
Drug	0.33	1.0
Router	0.43	0.60
Radio	0.38	0.83

Table 10. Aspects implying positive opinions

Product Name	Precision	Recall
Mattress	0.56	0.72
Drug	0.67	0.86
Router	0.40	1.00
Radio	0	0

Table 11. Aspects implying negative opinions

From Tables 9 - 11, we observe that the precision of the proposed method is still low, although the recalls are good. To better help the user find such words easily, we rank the extracted aspect candidates. The purpose is to rank correct noun aspects that imply opinions at the top of the list, so as to improve the precision of the top-ranked candidates. Two ranking methods are used:

1. Rank based on the statistical score Z in Equation (8). We denote this method with Z-rank.

2. Rank based on negative/positive sentence ratio. We denote this method with R-rank.

Tables 12 and 13 show the ranking results. We adopt the rank precision, also called the precision@N, metric for evaluation. It gives the percentage of correct noun aspects implying opinions at the rank position N. Because some domains may not contain positive or negative noun

aspects, we combine positive and negative candidate aspects together for an overall ranking for each dataset.

	Mattress	**Drug**	**Router**	**Radio**
Z-rank	0.70	0.60	0.60	0.70
R-rank	0.60	0.60	0.50	0.40

Table 12. Experimental results: Precision@10

	Mattress	**Drug**	**Router**	**Radio**
Z-rank	0.66		0.46	0.53
R-rank	0.60		0.46	0.40

Table 13. Experimental results: Precision@15

From Tables 12 and 13, we can see that the ranking by statistical value *Z* is more accurate than negative/positive sentence ratio. Note that in Table 13, there is no result for the Drug dataset because no noun aspects implying opinions were found beyond the top 10 results because there are not many such noun aspects in the drug domain.

3.5. Identifying Resource Term

In this section, we would introduce how to identify resource terms in opinion documents, which is another special and important aspect in opinion documents.

3.5.1. Introduction

Opinion mining or sentiment analysis based only on opinion words (e.g. "*bad*", "*good*") is far from sufficient. We introduce the noun product aspects that imply opinion in the previous section. There are still many other types of expressions that do not bear sentiments on their own, but when they appear in some particular contexts, they imply sentiments. In (Liu, 2010), several such expressions and their corresponding opinion/sentiment rules are introduced. We believe that all these expressions have to be extracted and associated problems solved before sentiment

1. Positive ← consume no or little resource
2. | consume less resource
3. Negative ← consume a large quantity of resource
4. | consume more resource

Figure 3. Sentiment polarity of statements involving resources.

analysis can achieve the next level of accuracy.

One such type of expressions involves resources, which occur frequently in many application domains. For example, *money* is a resource in probably every domain ("*this phone costs a lot of money*"), *gas* is a resource in the car domain, and *ink* is a resource in the printer domain. If a device consumes a large quantity of resource, it is undesirable. If a device consumes little resource, it is desirable. For example, the sentences, "*This laptop needs a lot of battery power*" and "*This car uses a lot of gas*" imply negative sentiments on the laptop and the car. Here, "gas" and "battery power" are resources, and we call these words *resource terms* (which cover both *words* and *phrases*).

In terms of sentiments involving resources, the rules in Figure 3 are applicable (Liu, 2010). Rules 1 and 3 represent normal sentences that involve resources and imply sentiments, while rules 2 and 4 represent comparative sentences that involve resources and also imply sentiments, e.g., "*this washer uses much less water than my old GE washer*". To the best of our knowledge, there is no reported algorithm that extracts resource terms. In this thesis, we propose an iterative algorithm to extract them from a domain corpus, e.g., a set of product reviews. In the above example sentence, we want to extract "*water*" as a resource term.

The related work is the general product aspect extraction (e.g., Hu and Liu, 2004, Popescu and Etzioni, 2005, Scaffidi *et al*. 2007, Titov and McDonald, 2008, Zhao *et al*., 2010). A resource in a domain is often an aspect or implies an aspect. For example, in "*this camera uses a*

lot of battery power", "*battery power*" clearly indicates *battery life*, which is an aspect of the camera. However, there are some important differences between resources and other types of aspects. The key difference is that resource terms often contribute directly to sentiments (e.g., based on the quantity that is consumed), while other aspects may not. e.g., "picture quality" in "*the picture quality of this camera is great*," where "great" solely determines the sentiment of the sentence. Thus, resource terms require special treatments in opinion mining or sentiment analysis.

We model the extraction problem with a bipartite graph and proposes a novel circular definition to reflect a special reinforcement relationship between *resource usage verbs* (e.g., *consume*) and resources (e.g., *water*) for resource extraction. We call the proposed method MRE (*M*utual *R*einforcement based on *E*xpected values). Based on the definition, the problem is solved using an iterative algorithm. To initialize the iterative computation, some global *seed resources* are employed to find and to score some strong resource usage verbs. These scores are applied as initialization for the iterative computation in the bipartite graph for any application domain. When the algorithm converges, we obtain a ranked list of candidate resource terms. Our experimental results based on 7 real-life data sets show the effectiveness of the proposed method. It outperforms 5 strong baselines.

3.5.2. Proposed Method

In this section, we present the proposed technique. Let us use the following two example sentences to develop the idea and the algorithm:

1. *This car uses a lot of gas*.
2. *This car uses less gas than Honda Civic*.

We call the first sentence a *normal sentence*, and the second sentence a *comparative sentence*.

From these two sentences, we can make the following observation:

Observation: The sentiment expressed in a sentence about resource usage is often determined by the triple,

(verb, quantifier, noun_term),
where *noun_term* is a noun or a noun phrase

In the first sentence, "uses" is the main verb, "a lot of" is a quantifier phrase, and "gas" is a noun representing a resource. In the second sentence, "uses" is also the main verb, "less" is a comparative quantifier, and "gas" is again a resource as a noun. We want to use such triples to help identify resources in a domain.

We notice that using only a pair, (verb, noun_term), or (quantifier, noun_term) is not sufficient. The pair (verb, noun_term) is unsafe because such pairs are very common since subject-verb-object is the most common English sentence structure, and the object is usually a noun term. Using (quantifier, noun_term) is also unsafe as the meaning of the noun terms following quantifiers can be diverse. By no means do we say that any above triple implies the last noun term is a resource. For example, "colors" is not a resource in "*this car got many colors*". The triples only find candidate resources, which need to be further analyzed.

Since it is unsafe to use the pair (verb, noun_term) or (quantifier, noun_term), we use only triples for candidate resource extraction. Due to the fact that it is easy to compile the main expressions of quantifiers, we just need to extract verbs and noun terms to discover candidate resources which are the noun terms. The quantifiers that we use in this work are listed in Table 14.

3.5.2.1.Extract Triples And Build a Graph

Since our algorithm is based on triples, we now discuss how to extract them. To extract triples from a corpus, part-of-speech (POS) tagging is first performed on each sentence. Verbs and nouns are then identified based on their POS tags. Verbs are words tagged as VB, VBD, VBZ, VBG, VBN, and VBP. Nouns are words tagged as NN and NNS. In addition, we regard a

Quantifiers
some, several, numerous, many, much,
more, most, less, least
a large/huge/small/tiny number of
a large/huge/small/tiny quantity/amount of
lot/lots/tons/ton/plenty/deal/load/loads of
[a] few/little

Table 14**:** A list of quantifiers

phrase with continuous POS tags of NN and NNS as a noun phrase, e.g., "spray/NN gel/NN" is seen as a single noun phrase "spray gel". In English grammar, quantifiers usually precede and modify noun terms. Thus, after locating a quantifier in a sentence, we extract its associated noun term, which directly follows the quantifier. After obtaining the noun term, we further exploit the dependency relation to find the associated verb in the sentence, since there is an assumed *verb-object* relationship between the verb and the noun. The relationship can be determined by a dependency parser. In our work, we approximate the dependency by making use of a text window in the sentence. It works quite well. Thus we did not use a dependency parser, which tends to be inefficient. We choose the closest verb in a text window (e.g., 10 words) before the noun as the verb part of the triple. Note that verbs such as "*is*", "*was*", "*am*" "*are*", "*were*" "*have*", "*has*", and

"*had*" are not used since they usually do not express resource usages. Finally, we lemmatize both the verb and the noun and store them only in the lemmatized format in a triple.

With all extracted triples, we build a bipartite graph based on the verb set V, the noun set N, and the set of links L between V and N. A link (i, j) is in L if there is a triple involving a verb $i \in V$ and a noun term $j \in N$. Note that in this graph, we do not use quantifiers, which are only used to identify candidate verbs and nouns.

3.5.2.2.Proposed Algorithm

We now present the proposed algorithm, which relies on the bipartite graph to encode a special kind of mutual enforcement relationship between resource usage verbs and resource terms. Before diving into the details of the algorithm, we define the following concepts.

Definition (Resource Term): A resource term represents a physical or virtual entity that can be consumed or obtained in order to benefit from it.

Some resources are general, which exist in many different application domains, i.e., "money" in "*this TV costs me a lot of money*". Other resources are more domain-specific, e.g., "onboard memory" in "*the phone uses more onboard memory*".

Definition (Resource Usage Verb): A resource usage verb (or *resource verb* for short) is a verb that can express resource usage.

Likewise, some resource verbs are general and can modify many different resource terms, e.g., "uses" in "*this car uses much more gas*", "*this washer uses a lot of water*", and "*this program uses a lot of memory*." Many others are more resource-specific, and tend to frequently co-occur with specific resources, e.g., "*spent*" in "*I spent too much money to buy the car*".

It seems that we can solve the problem of extracting resource terms using a simple graph propagation strategy. That is, given an application domain corpus, the user first provides a few seed resource terms. Using the bipartite graph, we can identify some resource verbs by following the links of the graph. The newly identified resource verbs are then used to identify new resource terms. The process continues until no more resource terms or verbs can be found.

However, this simple strategy has some major problems. First, as many resource verbs and terms are domain-specific, asking the user to provide some seeds for each domain is non-trivial. Second, many nouns (or verbs) in the triples may not be resources (or resource usage verbs), e.g., "*this car comes with many colors*." Any error resulted in the propagation can generate more errors subsequently.

With these concerns in mind, we propose a more sophisticated iterative algorithm. To solve the first problem above, we take a global approach. Instead of asking the user to provide some seed resources for each domain, we simply provide some global resource seeds, e.g., *water*, *money*, and *electricity*. Then in each application, the user does not need to do anything. Using these global resource seeds, we want to identify some good resource usage verbs. These verbs act as the initialization for the discovery of additional resource terms in each domain based on the domain corpus. The proposed method thus consists of two main stages. The first stage is only done once and the results are used for individual application domains as the initialization.

Stage 1: Identifying Global Resource Verbs

Global resource verbs are those verbs that can express resource usage of many different resources, e.g., *use* and *consume*. We can use a bipartite graph constructed from a large data set to find them. The following observations help us formulate the solution:

1. A global resource verb has links to many different resource terms. The more diverse the resource terms that a verb can modify, the more likely it is a good global resource verb.

2. Conversely, the more global resource verbs a resource term is associated with, the more likely it is a genuine resource term.

These two observations indicate that the global resource verbs and the resource terms have a mutual enforcement relationship, which can also be modeled by the Web page ranking algorithm HITS exactly, which has been introduced in Section 3.2.

In our scenario, global resource verbs act as hubs and resource terms act as authorities. We provided a list of common resources (seeds) (see Section 3.5.3.1). Using these seeds, we extract triples from the corpus and produce a link graph. The noun term set N consists of only these seed resource terms, and the V set consists of only those verbs which form triples with the N set. HITS is then applied on the graph. After HITS converges, each candidate resource verb has a hub score. We normalize them to the 0-1 interval. The resulting values are used to initialize the system for discovering resource terms from each application domain. That is, we do not need to execute stage 1 anymore.

Stage 2: Discovering Resource Terms in a Domain Corpus

Given the global resource verb values from stage 1 and a domain corpus, the stage 2 system identifies resource terms from the domain corpus.

In this stage, we still start with a bipartite graph as in the first stage. The graph can be constructed as discussed above by extracting triples from the domain corpus. On one side of the bipartite graph, it is the set of candidate resource terms N (noun terms) and on the other side, it is the set of candidate resource (usage) verbs V. For each $i \in V$, we want to compute its likelihood

of being a resource verb, denoted by $u(i)$, and for each noun term $j \in N$, we want to compute its likelihood of being a resource term, denoted by $r(j)$. If i and j are in a triple, a link (i, j) is in the link set L.

An obvious question is: Can we use HITS here as in stage 1? The answer is no. Unlike stage 1, the N set here is no longer a set of true resources, but only a list of noun terms, which are just candidate resources. A verb modifying multiple noun terms does not necessarily indicate that the verb is a resource usage verb. For example, it could be a general verb like "get". Also, as mentioned earlier, it is not always the case that if a noun term is modified by many verbs, it is a resource term. For example, it could be a topic word like "car" for the car domain. Applying the simple reinforcement relation in HITS is ineffective as we will see in the experiment section. To introduce the proposed technique, we make the following observations:

1. If a noun term is *frequently* associated with a verb (including quantifiers), the noun term is more likely to be a genuine resource term.
2. If a verb is *frequently* associated with a noun term (including quantifiers), it is more likely to be a genuine resource verb.

These two observations indicate that we should take verb and noun term co-occurrence frequency into consideration, which cannot be used in HITS. To consider frequency, we turn the frequency into a probability and make use of the expected value to compute scores for the verbs and noun terms, rather than summation in HITS.

In probability, given a random variable X, its *expected value* is defined as

$$E[X] = \sum_i p_i x_i \qquad (9)$$

where x_i is a possible outcome of the random variable X and p_i is the probability of x_i.

For our case, we have the following definitions for $u(i)$ and $r(j)$.

$$u(i) = \sum_{(i,j)\in L} p_{ji} r(j) \qquad (10)$$

where

$$p_{ij} = \frac{c(i,j)}{\sum_{(k,j)\in L} c(k,j)} \text{ and}$$

$$p_{ji} = \frac{c(i,j)}{\sum_{(i,k)\in L} c(i,k)}$$

$$r(j) = \sum_{(i,j)\in L} p_{ij} u(i) \qquad (11)$$

$c(i, j)$ is the frequency count of the link (i, j) in our corpus. p_{ij} is thus the probability of link (i, j) among all links from different verbs i to a noun j. p_{ji} is the probability of link (i, j) among all links from different nouns j to a verb i. We called this proposed algorithm MRE (*M*utual *R*einforcement based on *E*xpected values).

3.5.2.3.Smoothing the Probability

Although the idea is reasonable, we found an important issue when computing expected values. If a noun term j occurs only once, and it is connected with a strong resource verb i, its ranking value becomes very high. Due to its low frequency, the expected value of $r(j)$ is just the value of $u(i)$. In many cases, the value may be even higher than some frequent noun terms, whose value may be reduced by being associated with some non-resource verbs. This situation is not desirable. Since for sentiment analysis application, we should rank those frequent resource terms at the top instead of the terms which only occur once in the corpus.

The problem is that the probabilities of verbs or nouns are not reliable due to limited data. In order to handle infrequent verbs or noun terms, we smooth the probabilities to avoid probabilities of 0 or 1. The standard way of doing this is to augment the count of each distinctive verb/term with a small quantity λ ($0 \leq \lambda \leq 1$) or a fraction of a verb or noun term in both the numerator and denominator. Thus any verb and noun term will have a smoothed probability as follows.

$$p_{ij} = \frac{\lambda + c(i,j)}{\lambda \mid N \mid + \sum_{(k,j)\in L} N(k,j)} \quad (12)$$

$$p_{ji} = \frac{\lambda + c(i,j)}{\lambda \mid V \mid + \sum_{(i,k)\in L} N(i,k)} \quad (13)$$

This is called the *Lidstone smoothing* (Lidstone's law of succession) (Lidstone, 1920). We use λ to 0.01, which performs well. In the equations, $|V|$ is the total number of verbs and $|N|$ is the total number of noun terms in the graph.

Note that with smoothing, the original bipartite graph becomes a complete bipartite graph. Each added link is given a very small probability as computed using Equations (12) and (13).

3.5.2.4. The Computation Algorithm

The computation algorithm for the proposed method MRE is given in Figure 4. Q is the set of verbs from stage 1, and G is the bipartite graph. To initialize the iterative computation, we assign the hub score from stage 1 to each verb $i \in V$ as its initial score $u^0(i)$ if i is in Q (line 1). If i is not in Q, $u^0(i)$ is given the minimum value of the hub scores of all verbs in Q (line 2).

Algorithm: MRE (Q, G)

Input: A global resource verb set Q with their hub scores computed from HITS in stage 1, and G is the bipartite graph
Output: a ranked list of candidate resource terms

1. $u^0(i) \leftarrow H(i)$ of verb i, if *verb* $i \in Q$
2. $u^0(i) \leftarrow \arg\min_{r \in Q}\{H(r)\}$, if *verb* $i \notin Q$
3. **Repeat** till convergence
4. $r^{n+1}(j) = \sum_{(i,j)\in L} p_{ij} u^n(i)$
5. $u^{n+1}(i) = \sum_{(i,j)\in L} p_{ji} r^n(j)$
6. normalize $r(j)$ and $u(i)$
7. Output the ranked candidate resource terms based on their $r(j)$ score values.

Figure 4: The proposed MRE algorithm

After this initialization, the algorithm proceeds iteratively until convergence. We will describe the convergence characteristic of the algorithm in following section.

Finally, we note that unlike HITS, which converges to the same hub and authority (steady-state) scores regardless the initialization. For MRE, the initialization makes a big difference as we will see in the evaluation section.

3.5.3. Experiments

We now evaluate the proposed MRE method. We first describe the data sets, evaluation metrics, and then the experimental results. We also compare MRE with 5 baseline methods.

3.5.3.1. Data Sets and Global Resource Seeds

We used seven (7) diverse data sets to evaluate our technique. These data sets were crawled from the Web. Table 15 shows the domains (based on their names) and the number of

sentences in each data set ("Sent." means the sentence). Each data set contains a mixture of reviews, blogs, and forum discussions about one type of product. We split each posting into sentences and the sentences are POS-tagged using the Brill's tagger (Brill, 1995). The tagged sentences are the input to our system MRE.

The global resource terms (resource seeds) used in the first stage of our method are: "gas", "water", "electricity", "money", "ink", "shampoo", "detergent", "room" "fabric softener", and "soap". In stage 1 of our algorithm, we used the combined data set of those in Table 15 to compute the hub scores for global resources usage verbs found to be associated with the resource seeds through some quantifiers.

3.5.3.2.Evaluation Metrics

We adopt the rank precision, also called precision@N metric for the experimental evaluation. It gives the percentage of correct resource terms (precision) at different rank positions. This is a popular method used in search ranking evaluation because one does not know all the relevant pages. This is also the case in our work as we do not know how many resource terms have been mentioned in each of the data set.

3.5.3.3.Baseline Methods

TF (Triple Frequency): This method finds all triples of the form (verb, quantifier, noun_term), and then ranks them according to their frequency counts. This basically corresponds to the methods used in (Hu and Liu 2004; Popescu and Oren, 2005; Zhuang et al. 2006; Qiu et al. 2011) as it combines the frequency and dependency patterns of the triples. This method is reasonable because many triples are indeed resource usage descriptions, and those more frequent ones (ranked high) are more likely to be genuine ones.

TFR (Triple Frequency Ratio): This method is similar to the above method but it divides TF by the number of pairs (verb, noun_term) with the same verb and the same noun term as in the triple. The reason for doing so is that such pairs are very common because subject-verb-object (SVO) is the most common English sentence structure, and object is usually a noun term. If the ratio of the occurrences of the triple is small, it may not be a resource usage description and then should be ranked low because sentences containing resources are usually talking about resource usages.

HITS: This method simply runs the HITS algorithm in the second stage for each data set. In this case, the global initialization is not useful as HITS will reach a steady state regardless of the initialization.

MRE-NI: Our MRE method without initialization by the global resource usage verbs.

MRE-NS: Our MRE method without the probability smoothing.

3.5.3.4. Results and Discussion

Tables 16-18 give the precision results for top 5, top 10, and top 20 ranked candidate resource terms. Each value in the last column gives the average precision for the corresponding row. We note that in Table 18, there are no results for “Paint” and “Printer” because no resources were found by any algorithm beyond top 10 as there are not many resources in these domains. It is also important to note that those resources that have been used as global seeds in stage 1 of our algorithm are not counted in the precision computation for the results in the tables. In other words, the discovered resource terms are all new. From the tables, we can make the following observations:

1. TF and TRF perform poorly. We believe the reason is that frequent triples or frequent triple ratio do not strongly indicate resource usages.

2. The performance of the HITS algorithm is also inferior. For only two data sets (out of 7), it performs similarly to MRE for the top 5 results. Its average results are all much worse than those of MRE.

3. Global resource verbs are very useful. As we can see, without using them (MRE-NI), the results are dramatically worse.

4. Probability smoothing also helps significantly. Without it, MRE-NS produces worse results consistently compared with MRE.

5. MRE is the best method overall. On average, it consistently outperforms every baseline method. Moreover, it does better than the 5 baseline methods on every data set at every rank position except for the data set "Printer" for the top 10 results, for which HITS is better.

From these observations, we can conclude that our proposed MRE algorithm is highly effective and it outperforms all 5 baseline methods.

3.5.3.5. Algorithm Convergence

In this sub-section, we show the convergence characteristic of the proposed MRE algorithm.

Figure 5 shows the convergence behavior of MRE for the car data set, where the *x*-axis is the number of iterations, and the *y*-axis is the difference of the average 1-norm values of the vector *r* and vector *u* in two consecutive iterations. We can see that the algorithm converges quite fast, i.e., in about 8 iterations. For other data sets, they behave similarly. All of them converge within 6-9 iterations. In all experiments, the algorithm stops when the 1-norm difference is less than 0.01.

Data	Car	Washer	Paint	Printer	Haircare	Mobile	TV
# of Sent.	56880	9997	1655	16314	29347	25354	23901

Table 15. Experimental data sets

Data sets	Car	Washer	Paint	Printer	Haircare	Mobile	TV	Ave.
TF	0.40	0.20	0.60	0.80	0.40	0.40	0.20	0.43
TFR	0.40	0.40	0.40	0.80	0.40	0.40	0.60	0.49
HITS	0.60	0.40	0.20	0.80	0.60	0.40	0.40	0.49
MRE-NI	0.20	0.80	0.20	0.60	0.60	0.60	0.80	0.54
MRE-NS	0.60	0.60	0.60	0.80	0.60	0.40	0.40	0.57
MRE	1.00	0.80	0.60	0.80	0.60	0.80	0.80	0.77

Table 16. Experimental results: Precision@5

Data sets	Car	Washer	Paint	Printer	Haircare	Mobile	TV	Ave.
TF	0.40	0.20	0.70	0.60	0.30	0.50	0.50	0.46
TFR	0.30	0.50	0.60	0.50	0.40	0.40	0.50	0.46
HITS	0.50	0.60	0.50	0.70	0.50	0.50	0.40	0.53
MRE-NI	0.30	0.80	0.40	0.40	0.30	0.70	0.60	0.50
MRE-NS	0.70	0.60	0.70	0.60	0.60	0.70	0.40	0.61
MRE	0.90	0.80	0.80	0.60	0.70	0.80	0.60	0.74

Table 17. Experimental results: Precision@10

Data sets	Car	Washer	Paint	Printer	Haircare	Mobile	TV	Ave.
TF	0.40	0.30			0.20	0.35	0.35	0.32
TFR	0.30	0.50			0.30	0.20	0.40	0.34
HITS	0.55	0.65			0.50	0.50	0.35	0.51
MRE-NI	0.30	0.70			0.45	0.50	0.45	0.48
MRE-NS	0.60	0.65			0.50	0.55	0.45	0.55
MRE	0.75	0.70			0.65	0.60	0.55	0.65

Table 18. Experimental results: Precision@20

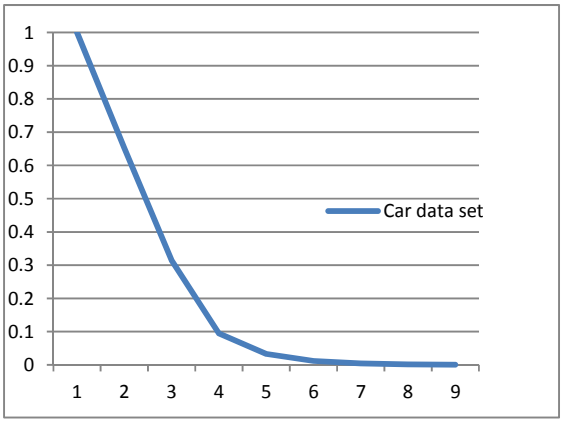

Figure 5. Convergent rate for car data

3.6. Conclusion

We proposed a new method to deal with the problems of the state-of-the-art double propagation method for aspect extraction. It first uses part-whole and "no" patterns to increase the recall. It then ranks the extracted aspect candidates by feature importance, which is determined by two factors: aspect relevance and aspect frequency. The HITS algorithm was used to measure aspect relevance. Experimental results using diverse real-life datasets show promising results.

We also proposed a method to identify noun product aspects that imply opinions. Conceptually, this work studied the problem of objective nouns and sentences with implied opinions. To the best of our knowledge, this problem has not been studied in the literature. This problem is important because without identifying such opinions, the recall of opinion mining suffers. Our proposed method determines aspect polarity not only by opinion words that modify the aspects but also by its surrounding context. Experimental results show that the proposed method is promising.

Moreover, we proposed the method to extract resource words and phrases in opinion documents. They are a class of terms that are important for opinion mining. When such resource terms appear with certain verbs and quantifiers, they often imply positive or negative sentiments or opinions. To the best of our knowledge, this work is the first attempt to discover such words and phrases. A novel iterative algorithm based on a circular definition of resource words and their corresponding verbs has been proposed. It was modeled on a bipartite graph and a special reinforcement relationship between resource usage verbs and resource terms. Experimental results based on 7 real-world opinion data sets showed that the proposed MRE method was effective. It outperformed 5 baseline methods in a large margin.

4. ENTITY EXTRACTION

4.1. Introduction

The problem of discovering entities is similar to the traditional named entity recognition problem. However, there is a major difference. In a typical opinion mining application, the user wants to find opinions on some competing entities, e.g., competing products or brands (e.g., *Canon*, *Sony* and many more). However, the user often can only provide a few names because there are so many different brands and models. Web users also write names of the products in various ways in forums and blogs. It is thus important for a system to automatically discover them from relevant corpora (e.g., blogs and forum discussions about cameras). The key requirement of this discovery is that the discovered entities must be relevant, i.e., they must be of the same class/type as the user provided entities. In the example above, the system should only discover camera brands and models.

This problem is the set expansion problem (Ghahramani and Heller, 2005), which expands a set of given seed entities. Formally, the problem is stated as follows: Given a set Q of seed entities of a particular class C, and a set D of candidate entities, we wish to determine which of the entities in D belong to C. That is, we "grow" the class C based on the set of seed examples Q. Clearly, this is a classification problem which needs a binary decision for each entity in D (belonging to C or not belonging to C). However, in practice, the problem is often solved as a ranking problem, i.e., to rank the entities in D based on their likelihoods of belonging to C. In our context, the user-given entities are the set of initial seeds. The system needs to expand the set using a text corpus. The classic methods for solving the problem are based on distributional similarity (Lee, 1999). This approach works by comparing the similarity of the word distribution

of the surround words of a candidate entity and the seed entities, and then ranking the candidate entities based on the similarity values. However, our results show that this approach is inaccurate. In this thesis, we present two new approaches to tackle this problem. The first one is based on a machine learning method called *positive and unlabeled* learning (PU learning) (Liu et.al, 2002). The other is based on a learning-based method called *Bayesian Sets* (Ghahramani and Heller, 2005), which estimates the probability that each candidate belongs to a "hidden" class represented by the seeds. The experiments performed on 10 real-life data sets from diverse domain shows good results.

4.2. Related Work

The proposed research is in the general area of information extraction, and more specifically named entity recognition (NER). There are extensive literatures on the topic, which can be grouped into three main categories: supervised learning, unsupervised learning and semi-supervised learning.

Supervised approaches are currently the dominant technique for solving the NER problem. Conditional Random Fields (CRF) is perhaps the most effective method (Lafferty et al, 2001). These methods need a large collection of labeled training examples. Labeling data is a labor-intensive and time-consuming process. Labeling also needs to be done for each domain. It is thus not suitable for applications that involve a large number of domains such as opinion mining. Our problem is also different as we have only a small number of seed entities. It is thus more related to semi-supervised learning.

Unsupervised approaches gather named entities from clustered groups based on the similarity of their contexts. This method usually relies on external knowledge (e.g., WordNet and

Wikipedia pages), class-specific extraction patterns (Banko et al, 2007), corpus-based term similarity(Pantel and Lin, 2002) and n-gram collocations (Downey et al, 2007) to find term clusters. Our work does not use the unsupervised method because we need a specific class of entities that are similar to the seeds provided by the user.

In semi-supervised approach, we have a set of seed entities. The system uses either class-specific patterns to populate entity term class or distributional similarity to find terms similar to the terms in the seed set. Our approach is semi-supervised as we also have a set of seeds.

Distributional similarity is the state-of-the-art technique for solving this problem. There are many distributional similarity measures. It is based on the hypothesis that words with similar meanings tend to appear in similar contexts. As such, a method based on distributional similarity typically fetches the surrounding context for each term (i.e. both seeds and candidates) and represents them as vectors by using TF-IDF or PMI (*Pointwise Mutual Information*) values (Lee, 1999; Lin, 1998). Similarity measures such as *Cosine*, *Jaccard*, *Dice*, etc, can then be employed to compute the similarities between each candidate vector and the seeds centroid vector (one centroid vector for all seeds). Lee (1999) surveyed and discussed various distribution similarity measures. We will show that this approach does not perform well for our data sets. It is able to find only those very frequently mentioned entities as they have more contextual information, but not those less frequently mentioned entities due to their weaker contextual information.

Positive and unlabeled learning (PU learning) and *Bayesian Sets* represents two different approaches which are based on machine learning. We will discuss them in detail in the next sections.

4.3. PU Learning Model

In machine learning, there is a class of semi-supervised learning algorithms that learns from *positive* and *unlabeled* examples (PU learning for short). The key characteristic of PU learning is that there is no negative training example available for learning. PU learning is a two-class classification model. It is stated as follows (Liu et al., 2002): Given a set P of positive examples of a particular class and a set U of unlabeled examples (containing hidden positive and negative cases), a classifier is built using P and U for classifying the data in U or future test cases. The results can be either binary decisions (whether each test case belongs to the positive class or not), or a ranking based on how likely each test case belongs to the positive class represented by P. Clearly, the set expansion problem can be mapped into PU learning exactly.

There are several PU learning algorithms (Liu et al, 2002; Li and Liu, 2003; Li et al, 2007; Yu et al, 2002). In this work, we used the S-EM algorithm given in (Liu, 2002). S-EM is efficient as it is based on naïve Bayesian (NB) classification and also performs well. The main idea of S-EM is to use a *spy* technique to identify some *reliable negatives* (*RN*) from the unlabeled set U, and then use an EM algorithm to learn from P, RN and $U–RN$.

The *spy* technique in S-EM works as follows (Figure 6): First, a small set of positive examples (denoted by *SP*) from P is randomly sampled (line 2). The default sampling ratio in S-EM is $s = 15\%$, which we also used in our experiments. The positive examples in *SP* are called "spies". Then, a NB classifier is built using the set $P– SP$ as positive and the set $U \cup SP$ as negative (line 3, 4, and 5). The NB classifier is applied to classify each $u \in U \cup SP$, i.e., to assign a probabilistic class label $p(+|u)$ (+ means positive). The probabilistic labels of the spies are then used to decide reliable negatives (*RN*). In particular, a probability threshold t is determined using the probabilistic labels of spies in *SP* and an input parameter l (noise level). Detailed explanation

can be found in (Liu et.al, 2002). t is used to find RN from U (lines 8-10). The idea of the spy technique is clear. Since spy examples are from P and are put into U in building the NB classifier, they should behave similarly to the hidden positive cases in U. Thus, they can help us find the set RN.

Given the positive set P, the reliable negative set RN and the remaining unlabeled set U–RN, an Expectation Maximization (EM) algorithm is run until convergence. In S-EM, EM uses the naïve Bayesian classification as its base method.

Algorithm Spy(P, U, s, l)

1. $RN \leftarrow \varnothing$; // Reliable negative set
2. $SP \leftarrow Sample(P, s\%)$;
3. Assign each example in $P - SP$ the class label +1;
4. Assign each example in $U \cup SP$ the class label -1;
5. $C \leftarrow$NB($P - S$, $U \cup SP$); // Produce a NB classifier
6. Classify each $u \in U \cup SP$ using C;
7. Decide a probability threshold t using SP and l;
8. **for** each $u \in U$ **do**
9. **if** its probability $p(+|u) < t$ **then**
10. $RN \leftarrow RN \cup \{u\}$;

Figure 6. Spy technique for extracting reliable negatives (RN) from U.

4.3.1. Data Preparaion

In this section, we are talking about how to prepare data from opinion documents for PU learning and S-EM algorithm.

Candidate entities: Since we are interested in extracting named entities, we select single words or phrases as candidate entities based on their corresponding part-of-speech (POS) tags. In particular, we choose the following POS tags as entity indicators — NNP (proper noun), NNPS (plural proper noun), and CD (cardinal number). We regard a phrase (could be one word) with a

sequence of NNP, NNPS and CD POS tags as one candidate entity (CD cannot be the first word unless it starts with a letter), e.g., "Windows/NNP 7/CD" and "Nokia/NNP N97/CD" are regarded as two candidates "Windows 7" and "Nokia N97".

Context: For each seed or candidate occurrence, the *context* is its set of surrounding words within a window of size *w*, i.e. we use *w* words right before the seed or the candidate and *w* words right after it. Stop words are removed.

Positive and unlabeled sets: For each seed $s_i \in S$, each occurrence in the corpus forms a vector as a positive example in *P*. The vector is formed based on the surrounding words context (see above) of the seed mention. Similarly, for each candidate $d \in D$ (see above; *D* denotes the set of all candidates), each occurrence also forms a vector as an unlabeled example in *U*. Thus, each unique seed or candidate entity may produce multiple feature vectors, depending on the number of times that it appears in the corpus. The components in the feature vectors are term frequencies for S-EM as S-EM uses naïve Bayesian classification as its base classifier.

4.3.2. Candidate Ranking

For distributional similarity, ranking is done using the similarity value of each candidate's centroid and the seeds' centroid (one centroid vector for all seeds). Rankings for S-EM are more involved. We discuss them below.

At convergence, S-EM produces a Bayesian classifier *C*, which is used to classify each vector $u \in U$ and to assign a probability $p(+|u)$ to indicate the likelihood that *u* belongs to the positive class. Recall that for S-EM, each unique candidate entity may generate multiple feature vectors, depending on the number of times that the candidate entity occurs in the corpus. As such, the rankings produced by S-EM are not the rankings of the entities, but rather the rankings of the entities' occurrences. Since different vectors representing the same candidate entity can have very

different probabilities (for S-EM), we need to combine them and compute a single score for each unique candidate entity for ranking.

To this end, we also take the entity frequency into consideration. Typically, it is highly desirable to rank those correct and frequent entities at the top because they are more important than the infrequent ones in applications. With this in mind, we define a ranking method.

Let the probabilities (or scores) of a candidate entity $d \in D$ be $V_d = \{v_1 , v_2 \ldots, v_n\}$ for the n feature vectors of the candidate. Let M_d be the median of V_d. The final score (*fs*) for d is defined as following:

$$fs(d) = M_d \times \log(1 + n) \qquad (14)$$

The use of the median of V_d can be justified based on the statistical *skewness* (Neter et al, 1993). If the values in V_d are skewed towards the high side (negative skew), it means that the candidate entity is very likely to be a true entity, and we should take the median as it is also high (higher than the mean). However, if the skew is towards the low side (positive skew), it means that the candidate entity is unlikely to be a true entity and we should again use the median as it is low (lower than the mean) under this condition.

Note that here n is the frequency count of candidate entity d in the corpus. The constant 1 is added to smooth the value. The idea is to push the frequent candidate entities up by multiplying the logarithm of frequency. *log* is taken in order to reduce the effect of big frequency counts.

The final score *fs*(d) indicates candidate d's overall likelihood to be a relevant entity. A high *fs*(d) implies a high likelihood that d is in the expanded entity set. We can then rank all the candidates based on their *fs*(d) values.

4.4. Bayesian Sets

Bayesian Sets is based on Bayesian inference, and was designed specifically for the set expansion problem (Ghahramani and Heller, 2005). The algorithm learns from a seeds set (i.e., a positive set *P*) and an unlabeled candidate set *U*. Although it was not designed as a PU learning method, it has similar characteristics and produces similar results as PU learning.

Candidate entities and their feature vectors (denoted by T) can be generated from the corpus similarly:

1. From the corpus, we identify all candidate entities and sentences that contain them.
2. For each candidate entity, a feature vector is produced based on all the sentences that contain the candidate entity.

The sets *R* and *T* are then used for Bayesian Sets learning. The learned model then assign a relevance score to each test vector representing a candidate entity. The score is then used to rank the candidate entities. The ranking is the final result.

Unfortunately, this direct application of Bayesian Sets produces poor results. We believe there are two main reasons. First, since Bayesian Sets uses binary features, multiple occurrences of an entity in the corpus, which give rich contextual information, is not fully exploited. Second, since the number of seeds is very small, the learned results from Bayesian Sets can be quite unreliable.

This technique can be directly applied to our context as follows. Given a set of seed entities and a text corpus, the seed data, a set of feature vectors (denoted by R) can be produced as follows:

1. A set of features is first designed to represent each entity.

2. For each seed entity, we identify all the sentences in the corpus that contain the entity. Based on the sentence contexts, we produce a feature vector to represent the seed entity.

We propose a more sophisticated method to use Bayesian Sets, which produces much better results. We introduce it here. Given a set of seed entities, the seed data are produced as follows:

1. Again, a set of features to represent each entity is designed.

2. For each seed entity, we identify all sentences in the corpus that contain the entity. From each sentence, a separate feature vector representing the seed entity in the sentence is produced. Hence, for each seed entity, we produce multiple vectors in the seed data. The number of vectors for the entity is the same as the number of sentences containing the entity.

Candidate entities and their data can be generated similarly:

1. From the corpus, all the candidate entities and sentences containing them are first identified.

2. For each candidate entity *e*, and for each sentence *s* that contains the entity, a feature vector is produced based on sentence *s* for entity *e*. Again, each candidate entity generates multiple feature vectors in the candidate data.

Clearly, the difference between this approach and the direct approach is that in the direct approach, each entity generates a single vector, while in the new approach each entity generates multiple vectors depending on the number of sentences containing the entity. Based on the generated data, we can run Bayesian Sets, and rank the feature vectors of the candidate entities.

This approach, however, causes two problems. The first problem is that since each candidate entity has multiple vectors and each vector has a different score produced by Bayesian Sets, the question is which score to use to rank the candidate entity. We need to decide this because the final result that we want is a ranked list of entities, not the vectors. This problem does not exist in the direct approach.

Before we deal with this problem, let us also describe an important issue that has not been addressed in the set expansion literature. The issue is the entity occurrence frequency. We consider an entity e_1 is more important than another entity e_2 if e_1 appears more frequently than entity e_2. In practice, it is desirable to rank those frequent entities higher than infrequent entities. The reason is that missing a frequently mentioned entity in opinion mining is very bad, but missing a rare entity is not a big issue. To deal with the two issues, we propose an effective method to combine both factors to rank entities with very good results.

The second problem is the feature sparseness. Due to the fact that each vector is only based on a single sentence that contains a candidate entity, the number of features contained in a sentence can be very small and even 0. Making best use of each feature becomes crucial since the score function learned from Bayesian Sets is essentially a weight vector corresponding to all features. Here again, we developed two techniques to remedy this situation. One of them is based on feature scaling and the other is based on automatically determining the quality of each feature, and then uses the feature quality information to re-weight each feature.

As with any supervised learning approach, for each application the user needs to design a new set of features. This report proposes a set of generic features for learning which have been shown to perform very well in diverse domains. This means that the user does not need to engineer features for each application.

Two more improvements were also made by enlarging the user-given seeds by using coordination patterns and by bootstrapping Bayesian Sets. All the proposed methods have been implemented and tested based on 10 real-life data sets (or text corpora) used for opinion mining. The data sets were collected from different review sites, user discussion forums and blogs on the Web.

4.4.1. Introduction

Let D be a collection of items and Q be a user-given *seed set* of items, which is a (small) subset of D (i.e., $Q \subseteq D$). The task of Bayesian Sets is to use a model-based probabilistic criterion to give a score to each item e in D (e $\in D$) to gauge how well e fits into Q. In other words, it measures how likely e belongs to the "*hidden*" class represented/implied by Q. Each item e is represented with a binary feature vector.

The Bayesian criterion score for item e is expressed as follows:

$$score(\mathrm{e}) = \frac{p(\mathrm{e}|Q)}{p(\mathrm{e})} \qquad (15)$$

$p(\mathrm{e}|Q)$ represents how probable that e belongs to the same class as Q given the examples in Q. $p(\mathrm{e})$ is the prior probability of item e. Using Bayes rule, the equation can be re-written as:

$$score(\mathrm{e}) = \frac{p(\mathrm{e},Q)}{P(\mathrm{e})P(Q)} \qquad (16)$$

Equation (16) can be interpreted as the ratio of the joint probability of observing e and Q, to the probability of independently observing e and Q. The ratio basically compares the probability that e and Q are generated by the same model with parameters θ, and the probability that e and Q are generated by different models with different parameters θ and $\widetilde{\theta}$. Equation (16) says that if the probability that e and Q are generated from the same model with the parameters θ is high, the

score of e will be high. On the other hand, if the probability that e and Q come from different models with different parameters θ and $\tilde{\theta}$ is high, the score will be low.

In pseudo code, the Bayesian Sets algorithm is given in Figure. 7.

Algorithm: BayesianSets(Q, D)

Input: A small seed set Q of entities
A set of candidate entities D (= $\{e_1, e_2, e_3 \ldots e_n\}$)
Output: A ranked list of entities in D

1. **for** each entity e_i in D
2. compute: $score(e_i) = \frac{p(e_i,Q)}{p(e_i)p(Q)}$
3. **end for**
4. Rank the items in D based on their scores;

Figure 7. The Bayesian Sets learning algorithm

If we assume that $q_k \in Q$ is independently and identically distributed (i.i.d.) and Q and e_i come from the same model with the same parameters θ, each of the three terms in Equation (16) are marginal likelihoods and can be written as integrals of the following forms:

$$p(Q) = \int \left[\prod_{q_k \in Q} p(q_k|\theta)\right] p(\theta) d\theta \quad (17)$$

$$p(e_i) = \int p(e_i|\theta) p(\theta) d\theta \quad (18)$$

$$p(e_i, Q) = \int \left[\prod_{q_k \in Q} P(q_k|\theta)\right] p(e_i|\theta) p(\theta) d\theta \quad (19)$$

Let us first compute the integrals of Equation (17). Each seed entity $q_k \in Q$ is represented as a binary feature vector (q_{k1}, q_{k2} , ... q_{kJ}). We assume each element of the feature vector has an independent Bernoulli distribution:

$$p(q_k|\theta) = \prod_{j=1}^{J} \theta_j^{q_{kj}} (1-\theta_j)^{1-q_{kj}} \quad (20)$$

The conjugate prior for the parameters of a Bernoulli distribution is the Beta distribution:

$$p(\theta|\alpha,\beta) = \prod_{j=1}^{J} \frac{\Gamma(a_j+\beta_j)}{\Gamma(a_j)\Gamma(\beta_j)} \theta_j^{\alpha_j-1}(1-\theta_j)^{\beta_j-1} \quad (21)$$

where α and β are hyperparameters (which are also vectors). We set α and β empirically from the data, $\alpha_j = km_j$, $\beta_j = k(1- m_j)$, where m_j is the mean value of jth components of all possible entities, and k is a scaling factor (we use 1 in our experiments and it works well). The Gamma function is a generalization of the factorial function.

For $Q = \{q_1, q_2, \ldots, q_N\}$, Equation (17) can be represented as follows:

$$p(Q|\alpha,\beta) = \prod_j \frac{\Gamma(\alpha_j+\beta_j)}{\Gamma(\alpha_j)\Gamma(\beta_j)} \frac{\Gamma(\tilde{\alpha}_j)\Gamma(\tilde{\beta}_j)}{\Gamma(\tilde{\alpha}_j+\tilde{\beta}_j)} \quad (22)$$

where $\tilde{\alpha}_j = \alpha_j + \sum_{k=1}^{N} q_{kj}$ and $\tilde{\beta}_j = \beta_j + N - \sum_{k=1}^{N} q_{kj}$. With the same idea, we can compute Equation (18) and Equation (19).

Overall, the score of e_i, which is also represented a feature vector, (e_{i1}, e_{i2} , ... e_{iJ}), is computed with:

$$score(e_i) = \prod_j \frac{\frac{\Gamma(\alpha_j+\beta_j+N)}{\Gamma(\alpha_j+\beta_j+N+1)} \frac{\Gamma(\tilde{\alpha}_j+e_{ij})\Gamma(\tilde{\beta}_j+1-e_{ij})}{\Gamma(\tilde{\alpha}_j)\Gamma(\tilde{\beta}_j)}}{\frac{\Gamma(\alpha_j+\beta_j)}{\Gamma(\alpha_j+\beta_j+1)} \frac{\Gamma(\alpha_j+e_{ij})\Gamma(\beta_j+1-e_{ij})}{\Gamma(\alpha_j)\Gamma(\beta_j)}} \quad (23)$$

We can simplify Equation (23) by using the fact that $\Gamma(x) = (x-1)\Gamma(x-1)$ for $x > 1$. In addition, for each j we can consider the two cases $e_{ij} = 0$ and $e_{ij} = 1$ separately. For $e_{ij} = 0$, we have a contribution $\frac{\alpha_j+\beta_j}{\alpha_j+\beta_j+N} \frac{\tilde{\beta}_j}{\beta_j}$. For $e_{ij} = 1$, we have a contribution $\frac{\alpha_j+\beta_j}{\alpha_j+\beta_j+N} \frac{\tilde{\alpha}_j}{\alpha_j}$. Putting above together, we have Equation (24).

$$score(e_i) = \prod_j \frac{\alpha_j+\beta_j}{\alpha_j+\beta_j+N} \left(\frac{\tilde{\alpha}_j}{\alpha_j}\right)^{e_{ij}} \left(\frac{\tilde{\beta}_j}{\beta_j}\right)^{1-e_{ij}} \quad (24)$$

The log of the score is linear in e_i:

$$\log score(e_i) = c + \sum_j w_j e_{ij} \quad (25)$$

where

$$c = \sum_j \log(\alpha_j + \beta_j) - \log(\alpha_j + \beta_j + N) + log\tilde{\beta}_j - log\beta_j$$

and $w_j = \log \tilde{\alpha}_j - log\alpha_j - log\tilde{\beta}_j + log\beta_j \quad (26)$

All possible entities e_i will be assigned a similarity score by Equation (25). Then we can rank them accordingly. The top ranked entities should be highly related to the seed set *Q* according to the Bayesian Set algorithm.

4.4.2. Candidate Entity and Entity Feature

In this section, we discuss how to generate candidate entities, i.e., the set *D*, and design features for Bayesian Sets learning. A template-based feature identification technique is proposed which requires no manual engineering of individual features. We also discuss the new method of data (feature vector) generation.

4.4.2.1. Candidate Entity Extraction

As the candidate entity extraction in PU learning method, we select a word as an entity candidate e (e ∈ *D*) based on its corresponding part-of-speech (POS) tags. Likewise, we regard a phrase with sequential POS tags of NNP, NNPS and CD as one entity candidate.

4.4.2.2. Template-Based Feature Identification

As noted above, each candidate entity in Bayesian Sets is represented as a binary feature vector. For an entity, if a feature value is 1, it means that the entity has the feature and 0 otherwise.

Like a typical learning algorithm, one has to design a set of features for learning. Our feature set consists of two subsets:

Entity word features (EWF): These features characterize the words representing entity themselves. This set of features is completely domain independent as we will see below.

Surrounding word features (SWF): These features are the surrounding words of a candidate entity.

Entity word features are about the makeup and characteristics of a word. They describe different word case combinations, and digits in words. We use the following word features:

EWF1: Only the first letter in the word is capitalized, e.g., Sony.
EWF2: Every letter in the word is capitalized, e.g., IBM.
EWF3: There is at least one letter in the word that is capitalized and it is not the first letter, e.g., iPhone.
EWF4: The word has a combination of letters and digits, e.g., N97.
EWF5: The word consists of only digits.

Regular expressions are used to detect the first 5 word features. Note that in this work we are interested in English text. We are aware that some features may not work well in other languages. For other languages, these features may need to be revised.

Surrounding word features are words on the left or right of the candidate entity. So we define the following six syntactic templates for feature extraction. The extracted features are specific words or phrases. In the templates below, EN refers to an entity/candidate. We use 5 words before and 5 words after the entity phrase/word as the text window. This window size works very well (but it can be changed).

Template 1: Left first verb of EN in the text window

For example, in the sentence, "*I have bought this EN LCD yesterday*", "bought" is extracted as a feature as it is the first verb on the left of EN.

Template 2: Left first noun of EN in the text window

For example, in the sentence "*I have taken 10 mg of EN*", "mg" is extracted as a feature as it is the first noun on the left of EN.

Template 3: Left first noun-preposition phrase of EN in the text window

For example, in the sentence, "*The performance of EN is awesome*", the phrase "performance of" is extracted as a feature as it is the first noun-preposition phrase on the left of EN. Note that the phrase is used as a feature.

Template 4: Right first verb of EN in the text window

For example, in the sentence "*This EN works*", "works" is extracted as a feature as it is the first verb on the right of the entity EN.

Template 5: Right first noun of EN in the text window

For example, in the sentence "*I went to a EN dealer*"; "dealer" is extracted as a feature as it is the first noun on the right of EN.

Template 6: Right first preposition-noun phrase of EN in the text window

For example, in the sentence, "*I am using EN for Arthritis*"; "for Arthritis" is extracted as a feature as it is the first preposition-noun phrase on the right of EN.

Note that noun here includes all forms of nouns, and verb here also includes all forms of verbs, they are determined by their POS tags. Stemming is also applied to reduce words to their stems.

For feature extraction, we only use the seed entities and the sentences in the given corpus that contain the seed entities. More specifically, we perform the following steps:

1. Identify all sentences that contain seed entities in the corpus.
2. For each sentence and for each entity, we use the above templates to match the left and the right contexts of the entity within the text window and extract all the matched words. These matched words are the pattern features.

Our experiments show the above templates perform very well in a wide range of domains, which is not surprising because the templates are so general and comprehensive that they basically cover all kinds of nearby context words except adjectives and adverbs. Adjective and adverbs are often non-specific. Any such word can modify almost anything, while in a domain some specific verbs and nouns are often associated with entities. Even we have this set of highly general templates, by no means, we claim/prove that these templates are sufficient for entity extraction in any domain. The approach of using templates, however, makes feature engineering much easier, compared to manually identifying words or phrases in each new domain.

4.4.2.3. Data Generation

Based on the seed entities Q and candidate entities D, and the extracted features, we can prepare data for learning. As we discussed above, we do not use the conventional way to generate the data, which generates one vector for each entity in the seed set or the candidate entity set

identified in the corpus, as it does not work well. Instead, for each appearance of an entity in a sentence, a feature vector is generated. Of course, this way of preparing the data causes the problems of feature sparseness and entity ranking.

4.4.2.4. Feature Reweighting

As we discussed in the introduction section, the way that we prepare the data causes serious data sparseness because each vector is only obtained from one sentence (either long or short) and because the features are only from sentences containing the given seeds. The number of pattern features in a vector can be very small or even 0. This requires best use of the features. We describe two techniques to improve feature weighting so that more features can be made effective and high quality features can be given more weights.

(1) Raising Feature Weights

From Equation (25), we see that the score of an entity e_i is determined only by its corresponding feature vector and the weight vector w = (w_1, w_2, …, w_J). Equation (26) shows a value of the weight vector w. We can rewrite Equation (26) as follows,

$$w_j = log\frac{\tilde{\alpha}_j}{\alpha_j} - log\frac{\tilde{\beta}_j}{\beta_j}$$

$$= \log\left(1 + \frac{\sum_{i=1}^{N} q_{ij}}{k m_j}\right) - \log\left(1 + \frac{N - \sum_{i=1}^{N} q_{ij}}{k(1-m_j)}\right) \qquad (27)$$

In Equation (27), N is the number of items in the seed set. As mentioned before, m_j is the mean of feature j of all possible entities and k is a scaling factor (which we use 1). m_j can be regarded as the prior information empirically set from the data.

In order to make a positive contribution to the final score of entity e, w_j must be greater than zero. Under this circumstance, we can obtain the following inequality based on Equation (27).

$$\sum_{i=1}^{N} q_{ij} > N m_j \qquad (28)$$

Equation (28) shows that if feature *j* is effective ($w_j > 0$), the seed data mean must be greater than the candidate data mean on feature *j*. Only such kind of features can be regarded as high-quality features in Bayesian Sets. Unfortunately, it is not always the case due to the idiosyncrasy of the data. There are many high-quality features, whose seed data mean may be even less than the candidate data mean. For example, in our drug data set, "prescribe" can be a left first verb for an entity. It is a very good entity feature. "Prescribe EN/NNP" (EN represents an entity, NNP is its POS tag) strongly suggests that EN is a drug. However, the problem is that the mean of this feature in the seed set is 0.024 which is less than its candidate set mean 0.025. So if we stick with Equation (28), the feature will have negative contribution, which means that it is worse than no feature at all. The fact that all pattern features are from sentences containing seeds, a candidate entity associated with a feature should be better than no feature.

We propose to tackle this problem by fully utilizing all features found by template patterns. We change original m_j to $\tilde{m}_j$ by multiplying a scaling factor t to force all feature weights $w_j > 0$:

$$\tilde{m}_j = t\, m_j \qquad (0 < t < 1) \qquad (29)$$

The idea is that we lower the candidate data mean intentionally so that all the features found from the seed data can be utilized. In other words, we let $\frac{\sum_{i=1}^{N} q_{ij}}{m_j}$ to be greater than *N* for all features *j*.

Since N is a constant, we only need to find the minimum value $\frac{\sum_{i=1}^{N} q_{ik}}{m_k}$ of all features, and lower m_k to $\tilde{m}_k$ to make $\frac{\sum_{i=1}^{N} q_{ijk}}{\tilde{m}_k} > N$. Under such condition, the parameter t for Equation (28) is determined by $\frac{\tilde{m}_k}{m_k}$. Note that although this may also make some bad features have $w_j > 0$, good features' weights will increase even more.

(2) Identifying High-Quality Features

Equation (27) shows that besides m_j value, w_j value is also affected by the sum $\sum_{i=1}^{N} q_{ij}$. It means that if the feature occurs more times in the seed data, its corresponding w_j will also be high. However, Equation (27) may not be sufficient since it only considers the feature frequency (as features are binary) but does not take feature quality into consideration. For example, we have two different features *A* and *B*, which have the same feature frequency in the seed data and thus the same mean, According to Equation (27), they should have the same feature weight w. However, for feature *A*, all feature count may come from only one entity in the seed set, but for feature *B*, the feature counts are from four different entities in the seed set. Obviously, feature B is a better feature than feature A simply because the feature is shared by or associated with more entities.

To detect such high-quality features to increase their weights, we use the following formula to change the original w_j to w_j' .

$$w_j' = r w_j \tag{30}$$

$$r = (1 + \frac{\log h}{T}) \tag{31}$$

In Equation (30), *r* is used to represent feature quality for feature *j*. *h* is the number of unique entities that have *j*th feature. *T* is the total number of entities in the seed set.

4.4.2.5. Entity Ranking

Although Bayesian Sets produces a rank of test vectors, due to the way that we generate the data (each entity produces multiple feature vectors depending on the number of times that it appears in the corpus), the ranking produced by Bayesian Sets is not a ranking of entities. Since different vectors representing the same entity can have very different scores, this leads to the problem of how to rank the entities, which is the final result that we need. Thus, we need to compute a score for each unique entity. Before we discuss further, let us describe another issue, the entity frequency. As mentioned in the introduction section, it is highly desirable to rank those correct and frequent entities at the top because they are more important than the infrequent ones in opinion mining (or even other applications). With this in mind, we define a new ranking algorithm.

Let the score values of a candidate entity e be $V = \{v_1 , v_2 \ldots, v_n \}$ obtained from the feature vectors representing the entity. Let M_d be the median value of V and M_m be the mean value of V. We use two steps to design the formula which is similar to the ranking method of PU learning method is Section 4.3.2.

Step1: Compute an entity score without considering frequency.

Let the score of a candidate entity e be S(e). We use the median as its score, i.e., S(e) = M_d. This is justified based on the statistical criteria *skewness*. We have discussed it in Section 4.3.2. The following are inequalities for skewness.

$$\begin{cases} M_m \geq M_d & if\ skewness \geq 0 \\ M_m < M_d & if\ skewness < 0 \end{cases} \quad (32)$$

where $skewness = \frac{m_3}{\sqrt{m_2}^3}$

$m_3 = \frac{\sum(v_i - \bar{v})^3}{n}$, $m_2 = \frac{\sum(v_i - \bar{v})^2}{n}$ and $\bar{v} = \frac{\sum v_i}{n}$.

If skewness < 0, we should use the median M_d. The intuition is that if the values in V are skewed towards the high side (negative skew), it means that the candidate entity is very likely to be a true entity, and we should take the median as it is also high. However, if the skew is towards the low side (positive skew), it means that the candidate entity is unlikely to be a true entity and we should again use the median as it is low under this condition.

Step2: Final score function considering the frequency

Factoring in the frequency, we have the final score function S'(e) for candidate entity e,

$$S'(\text{e}) = S(\text{e}) log(1 + f(\text{e})) \quad (33)$$

where f(e) is the frequency count of entity e. 1 is added to smooth the value. The idea is to push the frequent candidate entities up by multiplying the log of frequency. Log is taken in order to reduce the effect of big frequency count numbers.

4.4.3. The Overall Algorithm

Finally, we put everything together to present the final algorithm that we use. We can still improve the algorithm in two ways.

1. Enlarging the seed set using some high precision syntactic coordination patterns.
2. Iteratively running the Bayesian Sets algorithm with bootstrapping.

4.4.3.1. Enlarge Seed Set

One obvious way to improve the results is to increase the number of seeds. However, to do this automatically we need a very high precision method so as not to introduce non-entities into the seed list. Extraction based on syntactic *coordination* patterns is such a method.

Coordination patterns are linguistic structures (e.g., "*and*", "*or*") which connect two or more entities together in a sentence. It is a strong indicator that entities are of the same type or class. However, it also subjects to some noise. For example, the linguistic structure such as "… *X*, *Y*, …" may occur often without implying the same class of *X* and *Y* , but introducing apposition or clarification instead. In order to maximally prevent noisy data from being included into the expanded seed set, we need to utilize this technique with care. We only use the following 5 coordination patterns. Again, EN refers to an entity name. '|' is the logical OR relation. '*' stands for zero or more. Additionally, we also require that EN be a proper noun or a phrase of all proper nouns.

P1	EN [or | and] EN
P2	from EN to EN
P3	neither EN nor EN
P4	prefer EN to EN
P5	[such as | especially | including] EN (, EN)* [or | and] EN

Some example sentences are, "*Nokia and Samsung do not produce smart phones*," "*Neither Nokia nor Samsung produce big screen phones*" and "*LCD Brands such as Sony, Philips and Samsung, …*"

The extraction works by matching these patterns. If anyone of the ENs in the patterns is a seed in the seed list, the other matched ENs are extracted. The process can be run iteratively as the newly discovered entities can be used to match the patterns again and find more entities. The process stops after no more new entities are found. This process usually stops in no more than 3 iterations.

Algorithm: IterativeBayesianSets(Q, D, T)

Input: A small seed set Q of user-provided entities;
A data corpus T containing a collection of documents;
A set of candidate entities D extracted from T using the method in Section 4.4.2.1.
Output: A ranked list of entities in $D - Q$

1 $Q_+ \leftarrow$ Expand Q using T and the method in Section 4.4.3.1;
2. $D' \leftarrow$ Prepare data (feature vectors) using the method in Section 4.4.2, i.e., each appearance of a candidate entity in $D - Q_+$ in a sentence in T forms a feature vector;
3 $Q'_+ \leftarrow$ Prepare data (feature vectors) using the method in Section 4.4.2, i.e., each appearance of an entity in Q_+ in a sentence in T forms a feature vector;
4 $R \leftarrow$ BayesianSets(Q'_+, D'); // in Figure 7
5 $E \leftarrow$ Rank entities in $D - Q$ based on the scores in R using the method in 4.4.2 i.e., all candidate entities in D are ranked except those in the original Q.
6 $A \leftarrow$ Pick up the top k entities in E; // we use $k = 5$
7. $Q^* \leftarrow Q_+ \cup A$;
// Bootstrapping Bayesian Sets below

8. **while** ($Q^* \neq Q_+$) **do** // New seeds are added
9. $Q_+ \leftarrow$ Expand Q^* using T and the method in Section 4.4.3.1;
10. $Q'_+ \leftarrow$ Collect all feature vectors for the entities in Q_+;
11. $R \leftarrow$ BayesianSets(Q'_+, D'); // // in Figure 7
12. $E \leftarrow$ Rank entities in $D - Q$ based on the scores in R using the method in 4.4.2, i.e., all candidate entities in D are ranked except those in the original Q.
13. $A \leftarrow$ Pick up the top k entities in E; // we use $k = 5$
14. $Q^* \leftarrow Q_+ \cup A$
15. **end while**

16. Output the final ranked list E

Figure 8. The Iterative Bayesian Sets learning algorithm

4.4.3.2. Bootstrapping Bayesian Sets

This strategy again tries to find more seeds, but using Bayesian Sets itself. Thus, we run the Bayesian Sets iteratively. At the end of each iteration, we pick up k top ranked entities (we use $k = 5$ in our experiments). Note that the original seeds provided by the user are not in the rank. This newly extract k entity set is unioned with the current seed set. The iteration ends if no new entity is added to the current seed list, e.g., the union is the same as the current seed set. The detailed algorithm is given in Figure 8. The algorithm is self-explanatory. We will not discuss it further.

4.5. Experiments

We now evaluate the two proposed approaches and compare them with the latest distributional similarity based method in (Pantel et al, 2009), and show the effects of individual techniques proposed in the paper. We also compare our Bayesian Sets results with those of two Web based set expansion systems, Google Sets (Google, 2008) and Boo!Wa! (Wang and Cohen, 2005).

Data Sets: 10 diverse data sets were used for evaluation. They were obtained from a commercial company that provides opinion mining services. The data were extracted from multiple online message boards and blogs discussing products and services. Table 15 shows the domains and the number of sentences in each data set. We split each post into sentences and the sentences are POS-tagged using the Brill's tagger (Brill, 1995). The tagged sentences are the input to our system.

Evaluation Metric: Since we do not have gold standard entity sets to compare with, regular evaluation metrics such as precision and recall are not suitable for our purpose. We adopt rank precision, also called the Precision@N metric, which is the percentage of correct entities among the top *N* entities in a ranked list. The metric is commonly used for set expansion evaluation

We now present the experimental results in Tables 16 and 17. Table 16 shows the precisions of the top 15 results (i.e., precisions @15) based on three seeds randomly selected from a set of entities given by the clients of the company (we study the effect of different numbers of seeds below). The last column shows the average of each row. Due to space limitations, we can only give the average results for top 30 and 45 in Table 17. In Table 16, the results of six methods are compared.

Dist.S-fq: It denotes the *distributional similarity* method with the entity frequency considered as in Bayesian Sets. Without considering the frequency, it gives poorer results (see **Dist.S** in column 2 of Table 17). We implemented the latest model in (Pantel, 2009), which represents each entity as a *Pointwise Mutual Information* vector, and the similarity is compared using the *cosine* measure. For a fair comparison, we also added our word features (Section 4.4.2), which give better results.

PU.SEM: It denotes the S-EM method of PU learning model, which is discussed in Section 4.3. Its window size for a context window *w* is 3.

BaS-1-vec: It denotes the method that each entity is represented as a single feature vector (the direct application of Bayesian Sets).

BaS-no-fw: It denotes the proposed method BaS-all (see below) only without using the two feature reweighting techniques discussed in Section 4.4.2.

BaS-no-se: It denotes the proposed method BaS-all only without seed expansion using coordination patterns and bootstrapping (Section 4.4.3), but the two feature reweighting methods are applied.

BaS-all: It denotes the proposed method that utilizes all the new techniques.

From Tables 16 and 17, we can draw the following conclusions.

1. The distributional similarity methods (**Dist.S** and **Dist.S-fq**) perform poorly. Bayesian Sets with all our enhancements (**BaS-all**) is better by 14-16%. Direct application of Bayesian Sets with one feature vector per entity (**BaS-1-vec**) is also weak. The reasons have been given in Section 4.4.

2. **PU.SEM** outperforms **Dis.S-fq** by about 11%. But **BaS-all** outperforms **PU.SEM**, especially for top 15 results.

3. Comparing **BaS-no-fw** and **BaS-1-vec**, we can see that the new way of applying Bayesian Sets (BaS-no-fw: each entity with multiple feature vectors) produces much better results.

4. From the results of **BaS-no-se** (feature reweighting is used but no seed expansion) and **BaS-no-fw**, we see that the two feature reweighting methods are very helpful.

5. Comparing **BaS-no-se** and **BaS-all** (Table 17), we observe that expanding seeds and bootstrapping Bayesian Sets (Section 4.4.3) improve the results too.

Data Set	Cars	Insurance	Drug	Stove	Mattre	Cellphone	Bluray	Vacuum	OnlineBanking	LCD
# of sent	2223	12456	1504	2506	13233	15168	7108	13521	17441	1783

Table 15. Experimental Data sets or Corpora

	Car	Insurance	Drug	Stove	Mattress	Cellphone	Blu-ray	Vacuum	Online	LCD	Avg
Distr.S-fq	0.800	0.800	0.733	0.866	0.733	0.400	0.266	0.866	0.466	0.200	0.613
PU.SEM	0.933	0.866	0.666	0.933	0.666	0.733	0.666	0.533	0.533	0.733	0.726
BaS-1-vect	0.800	0.800	0.600	0.866	0.666	0.400	0.266	0.866	0.466	0.466	0.620
BaS-no-fw	0.800	0.733	0.666	0.833	0.733	0.600	0.533	0.933	0.600	0.800	0.723
BaS-no-se	0.866	0.733	0.733	0.866	0.766	0.533	0.600	0.933	0.666	0.733	0,743
BaS-all	0.866	0.800	0.800	0.866	0.800	0.666	0.600	0.933	0.666	0.800	0.779

Table 16. Precision @ 15 (3 seeds)

	Distr.S	Distr.S-fq	PU.SEM	BaS-1-vect	BaS-no-fw	BaS-no-se	BaS-all
Precision @ 15	0.553	0.613	0.726	0.620	0.723	0.743	0.779
Precision @ 30	0.536	0.543	0.650	0.540	0.620	0.643	0.686
Precision @ 45	0.495	0.493	0.617	0.510	0.568	0.597	0.626

Table 17. Average precision @ 15, 30, and 45 (3 seeds)

	2	3	4	5
Precision@15	0.760	0.779	0.785	0.785
Precision@30	0.666	0.686	0.692	0.692
Precision@45	0.611	0.626	0.626	0.633

Table 18. Effects of the number of seeds on precision

	Precision@15	Precision@30	Precision@45
BaS-all	0.779	0.686	0.622
Google Sets	0.692	0.609	0.560
Boo!Wa!	0.686	0.599	0.511

Table 19. Average precisions of BaS-all, Google Sets, and Boo!Wa! (3 seeds)

4.6. Effect of Seed Size

Table 18 gives the results of 2, 3, 4, 5 seeds. With more seeds, we can achieve better results. However, from the table, we observe that the improvements are minor. The reason is that

we have already applied the seed set enlargement step in the algorithm using coordination patterns and bootstrapped Bayesian Sets.

4.7. Compare With Google Sets and Boo!Wa!

Strictly speaking Google Sets (http://labs.google.com/sets) and Boo!Wa! (http://boowa.com/) are not comparable with our work as they mainly used Web pages structures to find lists, and they both use the "whole" Web to expand the seeds set. We are only interested in entities discussed in a specific opinion corpus, not things on the Web. For example, for cars, Google Sets returns major brands of cars, but does not find many specific models. It is thus of limited use for our purpose. Also, Google Sets mainly finds those official names of entities but in our corpora people use all types of short forms. For example, Motorola may be written as Moto and Samsung may be written as Sammy. Product model names have even more variations. Despite these, our method still performs better than Google Sets and Boo!Wa! (see Table 19). On average over the 10 seed sets (each has 3 seeds), our system outperforms both Google Sets and Boo!Wa! by a large margin.

5. TOPIC OPINION DOCUMENT EXTRACTION

Generally, for an opinion mining system, it first collects all blogs, forum discussion posts and reviews, and indexes them. When a user wants to study opinions on a type of products, keyword search is used to find relevant opinion documents for analysis. However, the documents that are retrieved in this way can have both low recall and low precision. As a result, the subsequent sentiment analysis step will not produce reliable results. An alternative method is to do supervised learning or classification. However, manual labeling of training data for each task is labor-intensive and time-consuming.

In this section, we propose a novel technique to solve this problem without the need of any manually labeled training data. Our experimental results show that the new method is highly effective.

5.1. Introduction

Nowadays, many large Web applications need to crawl and index web documents to be used later for different purposes or tasks. Typically, in such an application, the documents are not well categorized because one does not know what the future tasks will be. For example, in sentiment analysis, a company usually first collects blogs, forum discussion and reviews from the Web and stores them. When a client wants to study opinions on a type of products, the company needs to find all opinion documents related to the type of products from their data store, which contains a mixed set of documents from a large number of topics.

The usual approach is keyword search, i.e., the user issues some keywords to retrieve the relevant documents. Keywords may also be combined with some Boolean operators. However, this approach can result in both low precision and low recall. The reason for low precision is that a document that contains the keywords is not necessarily relevant. For example, if we want to study opinions on TVs and use the word "TV" to collect relevant reviews, we may retrieve many irrelevant documents such as "PS3" and "home theater" because they can contain the word "TV". The reason for low recall is that many documents that do not contain the word "TV" may be TV related documents.

Alternatively, we can model the problem as a traditional text classification problem (Sebastiani, 2002). The user manually labels some relevant and irrelevant documents for training. However, manual labeling is labor-intensive and time-consuming. For businesses, this means high costs.

In this thesis, we propose a novel approach to solve the problem. Instead of asking the user to label a large set of training documents, we only require him/her to provide a topic word, which is then used to retrieve some relevant documents. As mentioned above, this set of documents can still have low recall and low precision. We then employ a machine learning model called learning from positive and unlabeled examples (PU learning for short) to improve the recall and precision. Unfortunately, this approach is still not sufficient because the initially retrieved documents using the user-given topic word often have too low precision for learning. We thus propose a method to improve the precision of the initially retrieved set. One way of doing it is to use additional keywords. However, the problem is what additional keywords to use. This is not a trivial task and it is hard to choose manually. If they are not chosen well, the precision may not be improved, or the precision is improved but the retrieved document set is too small for accurate learning. We will propose a method to solve this problem.

In our experiments, we also found that the current main PU learning methods do not perform well in our scenario. The main reason is that our data does not fit their model assumptions. We thus propose a new PU learning method which is entirely different from the current state-of-the-art PU learning algorithms. Our experiments conducted using real-life document sets show that the proposed method is highly effective and can produce significantly better classification results.

5.2. Related Work

The proposed research is in the general area of text classification. There are extensive literatures on the topic, which can be grouped into three main categories, supervised learning, unsupervised learning and semi-supervised learning.

Supervised approaches are currently the dominant techniques for solving the text classification problem. To build a text classifier, a set of training documents is first labeled with predefined classes, and then a machine learning algorithm (e.g. Naïve Bayes (NB) (Lewis and Gale, 1994; McCallum and Nigam, 1998), or Support Vector Machine (SVM) (Joachims, 1998, 1999) is applied to the training examples to build a classifier. The trained classifier is then employed to assign class labels to the documents in the test set. As discussed in the introduction, labeling training data is costly and even impractical in practice. Our technique is different. We only ask the user to provide one word for the topic of his/her interests.

Unsupervised approaches (Jain and Dubes, 1988; Beil et al., 2002) are often used for text clustering which groups text documents into different categories. It can also be employed as a classifier. We can classify a test document to a cluster which is closest to it. However, the approach is generally less accurate than supervised learning.

In order to alleviate the burden of manual labeling, semi-supervised learning has been studied. It includes two main learning paradigms: The first paradigm is learning from labeled and unlabeled examples (LU learning for short) (e.g., Nigam et al., 1998, 2000). In this learning setting, there is a small set of labeled examples of every class, and a large set of unlabeled examples. The objective is to make use of the unlabeled examples to improve learning, which is different from our scenario as we do not want the user to label any training data. The second paradigm is learning from positive and unlabeled examples (PU learning for short) (Liu et al., 2002). This kind of learning assumes two-class classification. It only needs a set of labeled positive examples and a set of unlabeled examples, but no labeled negative examples. PU learning can model our problem well, since it is hard to obtain representative negative training data in our scenario. However, the existing PU learning methods do not produce satisfactory results. We will explain why in Section 5.3.2. Therefore, we adopt PU learning but propose a new approach.

The proposed work is also related to keyword-based methods for text classification. They are more appealing in practical settings since the user only needs to provide some keywords for each class. Liu et al. (2004) proposed a technique which asks users to label words for each class and use the words to perform feature selection and to train an EM classifier. Ko and Seo (2004) employed term similarity techniques to expand the seed keywords. Gliozzo et al. (2005) measured similarity between user-provided class names and documents in the Latent Semantic space (LSA). These works are different from ours as they work in the traditional classification context which requires training data for every class. We only automatically label some positive examples. Their methods may not applicable to our case because our negative class contains a large number of unknown topics. Our technique works by expanding a user-provided topic keyword and use the keywords to extract some initial training examples for classifier building using positive and unlabeled examples.

5.3. Proposed Approach

Formally, our problem can be defined as follows. Given a keyword q of a topic T of interest and a mixed set of unlabeled documents D of multiple topics, we want to identify or extract all documents in D that belongs to topic T. Our proposed approach consists of two steps:

Step 1: Obtain some initial positive training examples using keyword search.

Step 2: Build a text classifier using PU learning.

Below, we present the two steps in detail.

5.3.1. Step 1: Obtain Some Initial Postive Training Eaxmple

The step uses keywords to collect training examples from the mixed set. As illustrated in the introduction section, the user needs to provide one keyword first, which is usually easy since the word is often the topic word if the user knows what he/she is interested in. However, using a single word to retrieve relevant documents suffers from low precision. If we use two or more keywords, it is possible to obtain a much more accurate document set (positive training data). However, we need to be concerned with both precision and recall, which means we want the resulting set to be clean (with as fewer irrelevant documents as possible and yet to be as large as possible). If the set is clean (high precision) but too small, it will not be sufficient for subsequent learning. If the set is large (high recall), but noisy, it will not be good either. Thus, a good balance of precision and recall is important. Obtaining good keywords becomes critical.

We propose a technique to find such additional representative keywords based on the user-provided word. In our work, we use only one additional word. Given a word w_t and the document set D, we rank all the other words w in D by applying the following equations:

$$V(w) = Count(w_t, w) * idf(w) \quad (34)$$

$$idf(w) = log \frac{|D|}{|\{d: w \in d\}|} \quad (35)$$

where $V(w)$ is the ranking score of word *w*, $Count(w_t, w)$ is the document count that w_t and *w* co-occur, *d* is a document in *D*, $idf(w)$ is the *inverse document frequency* of word *w* in *D*. The additional word for search can be selected manually or automatically based on the ranking. In our experiments, it is selected automatically, which is the top-ranked word.

The reason that we use these equations is as follows: The additional representative word is supposed to be highly related to the user-provided word. $Count(w_t, w)$ measures the mutual relatedness between the two words. If the two words occur together frequently in the documents, they are likely to be highly related. However, that is not good enough. We need to take care of general or domain-specific stopwords (e.g. "Pros" and "Cons" in product reviews). Such words occur in almost every document. We thus use inverse document frequency $idf(w)$ to measure word importance in the document set, which is widely used in information retrieval. General and domain-specific stopwords are supposed to have small *idf* values. Equation (34) is thus a balanced measure for finding additional keywords. It is worth mentioning that we do not use the classic statistical *mutual information* or *pointwise mutual information* (PMI) measures, which are commonly used to measure the mutual relatedness of two items. The problem with them is that the top-ranked words can be rare words. Then, we will only obtain a very small positive set (low recall), which is insufficient for accurate learning in the next step. Another possible solution is to use feature selection (Liu et al., 2004). Again, the top-ranked words can be rare too.

In our experiments, the relevant documents extracted by two keywords have fairly good precision and reasonable recall. We then use PU learning to extract the rest of the positive documents in the mixed set.

5.3.2. Step 2: Classification Using PU Learning

5.3.2.1 The PU Learning Model

Learning from positive and unlabeled examples (PU learning for short) was originally proposed to solve the learning problem where no labeled negative training data exists. Formally, it is stated as follows: Given a set P of positive documents and a set U of unlabeled documents, which contains both (hidden) positive documents and (hidden) negative documents, a classifier is built using P and U that can identify positive documents in U or in a separate test set. Note that set U can be used in both training and testing because U is unlabeled. PU learning has been investigated by several researchers in the past decade. A theoretical study of Probably Approximately Correct (PAC) learning from positive and unlabeled examples was done in (Denis, 1998). Liu et al. (2002) proposed another theoretical foundation for PU learning. It shows that PU learning can be posed as a constrained optimization problem. Several practical PU learning algorithms have also been developed in recent years, such as S-EM (Liu et al., 2002), PEBL (Yu et al., 2002), Roc-SVM (Li et al., 2003), etc.

S-EM is a representative PU learning algorithm based on naïve Bayesian classification (NB) and the Expectation Maximization (EM) (Dempster et al., 1977) algorithm. It first uses a *spy* technique to identify some *reliable negative examples* (*RN*) from the unlabeled set U and then uses the EM algorithm to learn a NB classifier from P, *RN* and U-*RN*. Since it makes use of NB as the basic classification algorithm, it has a major advantage in practical applications due to NB's *efficiency* and *noise robustness*. However, in our experiments, we found this method did not perform well in many cases. The main reason is that the mixed data does not fit NB's assumption,

which we discuss below. As we will see in the experiment results section, the SVM-based method Roc-SVM also does not do well. The main reason is that the training data obtained from step 1 can still be noisy. SVM usually requires reasonably clean training data. For noisy data, NB is usually robust. In this paper, we propose a new PU learning algorithm based on an adjusted NB algorithm.

5.3.2.2. Naïve Bayesian Classification

Naïve Bayesian (NB) is one of the common and well-studied techniques for text classification (McCallum and Nigam, 1998). It has been shown to perform very well in practice by many researchers. It is a very popular classifier as it is both efficient and effective (McCallum and Nigam, 1998; Domingos and Pazzani, 1997; Ng and Jordan, 2002; Rennie et al., 2003).

NB is based on a generative model which models text documents as a mixture of multinomial distributions. A document is treated as a sequence of words and it is assumed that each word is generated independent of every other word given the class. We use $w_{d_{i,k}}$ to denote the word in position k of document d_i, where each word is from the vocabulary $V = < w_1, w_2, \ldots w_{|v|} >$, which is the set of all words we consider in classification. We also have a set of pre-defined classes, $C = \{c_1, \ldots, c_{|c|}\}$ (in our case, we only have two classes, $C = \{c_p, c_n\}$, where c_p is the positive class and c_n is the negative class). To perform classification, we need to compute the posterior probability $Pr[c_j|d_i]$, where c_j is a class and d_i is a document. Based on the Bayesian probability and the multinomial model, we have

$$Pr[c_j] = \sum_i Pr[c_j|d_i] / |D| \quad (36)$$

and with *Laplacian* smoothing

$$Pr[w_t|c_j] = \frac{1 + \sum_{i=1}^{|D|} N(w_t, d_i) P(c_j|d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N(w_s, d_i) P(c_j|d_i)} \quad (37)$$

where $N(w_t, d_i)$ is the number of times the word w_t occurs in document d_i and $Pr[c_j|d_i] \in [0,1]$. Finally, assuming that the probabilities of the words are independent given the class, we obtain the naïve Bayesian classifier:

$$Pr[c_j|d_i] = \frac{Pr\,[c_j] \prod_{k=1}^{|d_i|} Pr\,[w_{d_{i,k}}|c_j]}{\sum_{r=1}^{|c|} Pr\,[c_r] \prod_{k=1}^{|d_i|} P(w_{d_{i,k}}|c_r)} \quad (38)$$

For the NB classifier, the class with the highest $Pr[c_j|d_i]$ is assigned as the class of the document.

Apart from the normal conditional independence assumption above, the NB classifier for text classification also makes the following assumption:

- The text documents are generated by a mixture of multinomial distributions. There is a one-to-one correspondence between the mixture components and classes.

It is known that the conditional independence assumption does not cause much problem for text classification, but the one-to-one correspondence assumption can be a major problem for EM using NB as the base classifier (Nigam et al., 2000). Intuitively, the assumption says that each class should come from a distinctive distribution (or topic) rather than a mixture of multiple distributions (or topics). In our scenario, this assumption is often severely violated. As we perform binary classification on the mixed data set, inevitably, NB would model documents of multiple topics as the negative class. This results in an unrepresentative model, and consequently poor classification results. To illustrate the problem, we introduce the decision boundary of NB derived from Equation (38). NB determines the class label of a test case using the value *E* below, i.e., if *E* is greater than zero, the document is classified as c_p (positive) and otherwise c_n (negative):

$$E = log \frac{Pr\,[c_p] \prod_{k=1}^{|d_i|} Pr\,[w_{d_{i,k}}|c_p]}{Pr\,[c_n] \prod_{k=1}^{|d_i|} Pr\,[w_{d_{i,k}}|c_n]}$$

$$= \sum_{k=1}^{|d_{i|}} log \frac{Pr\,[w_{d_{i,k}}|c_p]}{Pr\,[w_{d_{i,k}}|c_n]} + log \frac{Pr\,[c_p]}{Pr\,[c_n]} \quad (39)$$

From Equation (39), we can see that NB is a linear classier in the *log* space. NB's poor performance for our mixed data set results from its decision boundary bias, which causes the classifier to unwittingly prefer one class over the other. Assume c_n^i is a topic in the negative class. w_t is a word in the document set of the topic and it is a good indicator for the topic c_n^i, i.e., $Pr[w_t|c_n^i] > Pr[w_t|c_p]$. Then w_t will make a positive contribution to class c_n^i if c_n^i is the only topic in the negative class c_n. However, in our case, the negative class c_n is mixed with many different topics. For other topics in c_n, w_t is likely to have a lower probability, i.e., $Pr[w_t|c_n^j] < Pr[w_t|c_n^i]$. However, according to Equation (37), we can easily infer that $Pr[w_t|c_n] < Pr[w_t|c_n^i]$ for the mixed data set. In some cases, $\Pr[w_t|c_n]$ is even less than $Pr[w_t|c_p]$. Then w_t changes to have more contribution to class c_p instead of class c_n, which is clearly wrong. This causes NB to prefer positive class incorrectly.

Since PU learning typically works iteratively (e.g. S-EM), if NB is used as the base classifier, its results can deteriorate with each additional iteration. This is the problem with S-EM.

5.3.2.3 A New PU Learning Algorithm

In this section, we introduce our new algorithm for PU learning to deal with the one-to-one correspondence problem. It is based on an adjusted NB and the EM algorithm. The proposed algorithm is thus called A-EM. It changes NB by adjusting word probabilities in the negative class, which helps to deal with the above problem. It still uses EM to iteratively increase recall and precision.

Word probability adjustment: As discussed above, the key problem with the one-to-one correspondence assumption is that the probabilities of representative words for the negative class are underestimated by NB. Thus, the proposed algorithm tries to compensate for that by using a "weighting" factor to increase their probabilities. The basic idea is that for a *good* word w_t in the negative class, we increase its probability towards the negative class, i.e., we increase $Pr[w_t|c_n]$; otherwise, we lower its probability in the negative class since it is likely to be a good word for the positive class c_p. We judge whether the word is good or bad for a class based on its original word probability. In the modified NB model, the document probability is determined by Equation (40), and the word probability is reweighted according to Equation (41).

$$Pr[c_n|d_i] = \frac{Pr[c_n]\prod_{k=1}^{|d_i|}\widetilde{Pr}[w_{d_i,k}|c_n]}{Pr[c_n]\prod_{k=1}^{|d_i|}\widetilde{Pr}[w_{d_i,k}|c_n]+Pr[c_p]\prod_{k=1}^{|d_i|}Pr[w_{d_i,k}|c_p]} \quad (40)$$

where

$$\widetilde{Pr}[w_{d_i,k}|c_n] = \begin{cases} k \times Pr[w_{d_i,k}|c_n] & if\ Pr(w_t|c_n) > Pr(w_t|c_p) \\ 1/k \times Pr[w_{d_i,k}|c_n] & if\ Pr(w_t|c_n) \leq Pr(w_t|c_p) \end{cases} \quad (41)$$

where k (>1) is a parameter and $w_{d_i,k} = w_t$. It is clear that k tries to increase the word probability for the negative class to deal with underestimation. In order to prevent the NB classifier from "pulling" positive examples into the negative side, we also lower the negative class probabilities of those words that are likely to be good for the positive class. k is adjustable (not fixed) in EM iterations. We will discuss how to set its value below.

Note that this approach is related to feature weighting for NB in (Zhang and Sheng, 2004; Kim et al., 2006, Frank et al., 2003; Hall, 2007), which gives higher weights to good features. However, they are different from our work as our purpose is not to find good features and increase their weights, but to decrease and increase the probability of words in the negative class

at the same time to deal with the one-to-one correspondence assumption of NB. $Pr\left[w_{d_i,k}\middle|c_p\right]$ is never changed.

EM algorithm: The EM algorithm is a class of iterative algorithm for maximum likelihood estimation in problems with missing data. Its objective is to estimate the values of the missing data (or a distribution over the missing values) using existing values. The EM technique as applied to our scenario with adjusted NB yields an effective algorithm. First, a NB classifier is built in the standard supervised fashion using the automatically retrieved documents from step 1 as positive examples and the rest of the unlabeled documents as negative examples. Classification is then performed on the test data with the adjusted NB classifier (Equation (40)). The test data is the original data set and each document is assigned a probability associated with each class. A new NB classifier is then built for the next iteration of EM using all the new probabilistic labels. We iterate this process until a stopping criterion is met (see below).

Tuning *k* using a performance measure: In Equation (41), parameter *k* plays an important role in determining word probability ratio of positive and negative classes. If it is chosen suitably, the modified NB can lessen the bias and obtain accurate class estimates. The question is how to choose *k*.

We now propose a method. Similar to the *Hill Climbing* (Russell and Norvig, 2003) approach, the proposed method searches for the best *k* iteratively. The technique starts with a big value for *k* (1.2 in our case). Then it attempts to maximize a classification measure by changing *k* iteratively. If the change produces a better result, it keeps *k* value. Otherwise, it keeps decreasing *k* in the next EM iterations until a stopping criterion is met (maximum number of iterations or *k* is close to 1). The *k* value for iteration *t*+1 (k_{t+1}) is computed with:

$$k_{t+1} = \begin{cases} k_t - \frac{(k_t - 1)}{h} & if\ m_{t+1} < m_t \\ k_t & if\ m_{t+1} \geq m_t \end{cases} \quad (42)$$

where h is a decaying factor (we use 3), which determines search step size. m_{t+1} is the classification measure value for the current EM iteration, m_t is the one for the previous EM iteration.

As discussed above, we set a large value for k initially to make the classification with high precision for the positive class in early steps. But it may cause low recall. Thus we lower k gradually to increase the recall. From the equation, we can see that the search step size for k is also changing. For initial EM iterations, the search step is large. With more and more unlabeled documents introduced in classification, we decrease the search step size to find more accurate classification result.

Now we discuss what the classification measure is. Although we have automatically retrieved positive examples, we have no negative examples, which mean that we cannot use the traditional precision, recall and F-score to compute the classification accuracy. Fortunately, this problem has been studied before in by Lee and Liu (2003). They proposed a measure which behaves similarly to F- score, and it can be estimated using a validation set. The measure is:

$$m = \frac{r^2}{Pr\,[f(d)=1]} \quad (43)$$

where $f()$ is the classifier and $\Pr(f(d)=1)$ is the probability that a document d is classified as positive, and r is recall. $\Pr(f(d) =1)$ can be estimated using the unlabeled data, while r can be estimated using a validation set randomly drawn from the automatically retrieved positive example set.

Algorithm: A-EM (*w* , *U*)

Input: key word *w*, unlabeled mixture data set *U*
Output: positive documents from *U*

1. $P \leftarrow \varnothing$; // positive training set
2. $P \leftarrow$ use keyword search to find positive training examples (section 5.3.1).
3. $PV \leftarrow sample(P , s\%)$ // positive validation set
4. Initially, assign each document $d_i \in (P - PV)$ the class probability $Pr[d_i|c_p] = 1$, which can change with iterations of EM below.
5. Initially, assign each document $d_j \in (U - P)$ the class probability $Pr[d_i|c_n] = 1$, which can change with iterations of EM below.
6. Run EM using *P–VP, U–P* and *U–PV* until it stops. In each EM iteration, build classifier $f \leftarrow NB(P - PV, U - P)$; classify each document in $(U - PV)$ using f; based on measure *m,* change weighting parameter *k* (section 5.3.2.3).
7. Decide the final classifier *F* with the highest *m value*.

Figure 9. The proposed learning algorithm

The overall algorithm for step 2 is given in Figure 9, which is self-explanatory and we will not explain it further.

5.4. Empirical Evaluation

This section evaluates the proposed approach A-EM. We compare it with the following baseline methods: (1) Supervised learning methods NB and SVM[2]. Their positive training data also comes from our keyword search as discussed in Section 5.3.1. The rest of the data is treated as negative training data. (2) PU learning methods S-EM and Roc-SVM, which are publically available from the download page of the LPU system[3]. (3) The proposed method without probability weighting for NB in Section 5.3.2.3.

[2]. http:// http://svmlight.joachims.org/
[3]. http://www.cs.uic.edu/~liub/LPU/LPU-download.html

5.4.1. Data Sets

We use four diverse document sets to evaluate our technique. They are product reviews of four companies crawled from the website "Google Product"[4]. Each document set is a set of mixed reviews of different products from a company or brand. To ensure the data quality, we manually inspected and determined the membership of each review document's product category (which is the topic used earlier). Tables 20 - 23 show the product names and their corresponding numbers of review documents.

TV	**Camera**	**Battery**	**PSP**
277	377	471	376
PS3	**Blueray**	**Headphone**	**Home theater**
376	147	435	309

Table 20. The Sony data set

Microwave	**Refrigerator**	**Washer**	**Dryer**
158	151	357	165

Table 21. The General Electric (GE) data set

TV	**Phone**	**Monitor**	**Blueray**	**Camera**	**Home theater**
317	205	214	327	245	155

Table 22. The Samsung data set

Zune	**Headset**	**Camera**	**Xbox**	**Mouse**	**Keyboard**
309	298	328	258	314	312

Table 23. Microsoft data set

5.4.2. Experiment Results

The experimental results are presented in Tables 25 - 28 and Figures 10 - 13. Given each data set, our objective is to identify all reviews for each category of product. Only two keywords were used to retrieve the initial positive training set. One was given (representing the product type), and the other was chosen by the method introduced in Section 5.3.1. The given words are

[4]. http://www.google.com/products

the data names in Tables 20 - 23. The evaluation metrics are F score, precision and recall. Six methods are compared. They share the same training data initially for fair comparison.

NB: Naïve Bayesian classifier.
SVM: Support Vector Machines. We used SVM^{light}.
S-EM: The S-EM classifier.
RVM: The Roc-SVM classifier.
I-EM: Our proposed method without probability weighting. (Nigam el al., 2000)
A-EM: The proposed classifier.

	h = 2	3	4	5
F score	0.780	0.796	0.799	0.801
Precision	0.838	0.846	0.847	0.846
Recall	0.776	0.784	0.787	0.790

Table 24. *h* values for Equation (42)

From Tables 25 - 28 and Figures 10- 13, we can draw the following conclusions:

1. SVM and Roc-SVM perform poorly. Although they can get very high precisions, their recalls are all very low. We believe the reason is because SVM is sensitive to noise. The positive training data obtained through keyword search in our scenario can be noisy.

2. S-EM performs well for simple data sets (e.g. GE data) when the one-to-one correspondence assumption is not badly violated. In this case, A-EM performs similarly. However, when the assumption is badly violated (the other three data sets), S-EM performs poorly and A-EM is significantly better than S-EM in F-score.

3. Directly applying NB or SVM is not a good idea, which shows that PU learning is useful.

Table 24 shows the average F-score results of A-EM over the four data sets by using different values of decay parameter *h* in Equation (42). It is true that a bigger *h* can make

	TV	Camera	Battery	PSP	PS3	Blueray	Headphone	Home theater	Ave
NB	0.728	0.886	0.794	0.886	0.791	0.462	0.726	0.564	0.728
SVM	0.712	0.403	0.229	0.644	0.495	**0.536**	0.132	0.178	0.378
S-EM	0.515	0.917	0.924	**0.900**	0.711	0.251	0.974	0.596	0.723
RVM	**0.786**	0.467	0.401	0.798	0.684	0.427	0.189	0.214	0.495
I-EM	0.523	0.936	0.934	0.782	0.662	0.378	0.951	0.617	0.722
A-EM	0.611	**0.947**	**0.939**	0.895	**0.804**	0.413	**0.955**	**0.742**	**0.788**

Table 25. F score for Sony data set

	Microwave	Refrigerator	Washer	Dryer	Ave
NB	0.494	0.719	0.541	0.586	0.585
SVM	0.233	0.368	0.311	0.386	0.324
S-EM	0.695	**0.872**	**0.872**	0.618	**0.764**
RVM	0.388	0.530	0.363	**0.671**	0.488
I-EM	0.714	0.797	0.780	0.475	0.692
A-EM	**0.718**	0.838	0.854	0.584	0.748

Table 26. F score for the GE data set

	TV	Phone	Camera	Monitor	Blueray	Hometheater	Ave
NB	0.813	0.736	0.554	0.409	0.641	0.330	0.585
SVM	0.766	0.254	0.287	0.281	0.467	0.253	0.384
S-EM	0.593	0.865	0.814	0.603	0.665	**0.428**	0.661
RVM	0.798	0.395	0.328	0.308	0.507	0.280	0.436
I-EM	0.733	0.897	0.824	0.599	0.701	0.338	0.686
A-EM	**0.811**	**0.900**	**0.826**	**0.660**	**0.788**	0.420	**0.734**

Table 27. F score for the Samsung data set

	Zune	Headset	Camera	Xbox	Keyboard	Mouse	Ave
NB	0.878	0.543	0.452	0.937	0.943	0.788	0.756
SVM	0.576	0.358	0.147	0.573	0.703	0.398	0.459
S-EM	**0.948**	0.600	0.791	0.957	**0.963**	0.891	0.858
RVM	0.687	0.452	0.202	0.658	0.764	0.476	0.539
I-EM	0.942	0.511	0.768	**0.958**	0.875	0.950	0.834
A-EM	0.946	**0.754**	**0.929**	**0.958**	0.954	**0.964**	**0.917**

Table 28. F score for the Microsoft data set

equation (42) search more values for k. However, from the experiments we find that the final classification

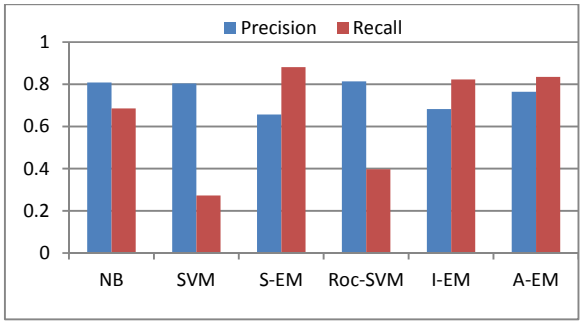

Figure 10. Precision and recall for Sony data

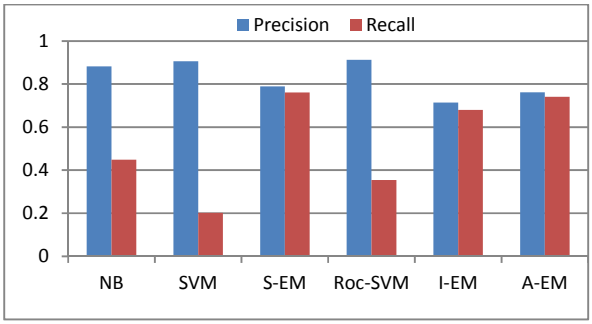

Figure 11. Precision and recall for GE data

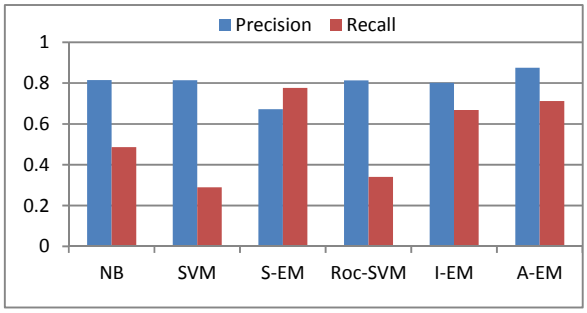

Figure 12. Precision and recall for Samsung data

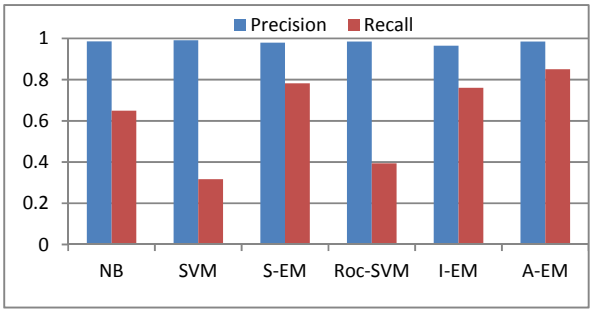

Figure 13. Precision and recall for Microsoft data

results are very close for different h values. We believe that the reason is that we often get optimal k value in a late EM iteration. In that case, search step is small enough no matter what h value is. Thus, h does not play a critical role for choosing k. We use $h = 3$ for faster convergence. For all our experiments, the algorithm runs 10-30 iterations. Note that the execution time is not an issue because NB is a linear algorithm, and A-EM simply runs NB 10-30 times.

In summary, we can conclude that the proposed A-EM technique is highly effective and it outperforms all 5 baseline methods.

6. CONCLUSION AND FUTURE WORK

This thesis studies aspect extraction and entity extraction problems, which are two essential components of an aspect-based opinion mining system. Without knowing aspects and entities that opinions are expressed on, the opinions are of limited use.

For aspect extraction, we propose an unsupervised method for general aspect extraction and ranking. It extracts product aspects from reviews based on dependency rules and language patterns. Meanwhile, it can rank them by importance. Besides general aspects, we discover that there are two special product aspects which are important for opinion mining. One is noun aspect implying opinion, the other is resource term. To the best of our knowledge, we are first to study these two new problems. Experiments show that the proposed methods are effective and promising.

For entity extraction in opinion documents, we regard it as a set expansion problem. The classic method of solving the problem is based on distributional similarity. This approach is effective in finding some very frequent entities but is weak for finding less frequent entities. The reason is that infrequent entities have less contextual information. In our work, we employ the PU learning and Bayesian Sets approach. However, directly applying Bayesian Sets gives poor results. We thus propose a more sophisticated way to use it, which, however, causes two major problems: entity ranking and feature sparseness. We have proposed some effective methods to solve the problem. Additionally, we also propose a set of generic features used for Bayesian Sets learning, which have been shown effective for a diverse set of domains. This is crucial for scaling-up opinion mining applications as it is too time-consuming to design features for each

individual application. Extensive experiments based on 10 data sets (forum discussions and blogs) show that the proposed method outperforms a recent distributional similarity method, the direct application of Bayesian Sets, Google Sets and Boo!Wa! by large margins.

We also study how to extract topic documents from a collection of opinion documents. It is also an important step in opinion mining. Without identifying correct topic opinion documents, the subsequent aspect extraction, entity extraction or sentiment analysis tasks will not produce reliable results. We propose a novel technique to solve this problem without the need of any manually labeled training data. It is based on a new PU Learning model. Experiments show promising results.

In our future work, for aspect extraction, we plan to study the problem of extracting and mapping implicit aspects that are verbs or verb phrases. In this thesis, we focus on extracting product aspects which are noun or noun phrases. However, it is known that many verbs or verb phrases can indicate implicit product aspects in opinion documents. For example, the sentence "*The refrigerator does not make ice*". The sentence expresses a negative opinion on the implicit aspect "ice function" of the refrigerator. How to extract such verb expression and map it to an implicit aspect is a challenging work. For entity extraction, we find that for the same entity in opinion documents, people may express it with many different words or phrases. For example, both "Mot phone" and "Moto phone" refer to the same entity "Motorola phone". To produce a useful opinion summary, these words, phrases or expressions should be grouped under the same entity group. However, Limited work has been done on clustering or grouping of synonym entities. It would be an interesting research direction in opinion mining.

CITED LITERATURE

Banko. M, M. Cafarella, S. Soderland, M. Broadhead and O. Etzioni. Open information extraction from the web. In *Proceedings of International Joint Conferences on Artificial Intelligence* (*IJCAI-2007*), 2007.

Beil, F., M. Ester, and X. Xu. Frequent term-based text clustering. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD-2002*), 2002.

Blair-Goldensohn, Sasha., Kerry. Hannan, Ryan. McDonald, Tyler. Neylon, George A. Reis, Jeff. Reyna. Building sentiment summarizer for local service reviews In *Proceedings of the Workshop of NLPIX in International Conference on World Wide Web* (*WWW-2008*), 2008

Blei, D., A. Ng, and M. Jordan. Latent dirichlet allocation. The Journal of Machine Learning Research, 2003.

Brill, E. Transformation-Based error-driven learning and natural language processing: a case study in part of speech tagging. Computational Linguistics, 1995.

Brody, S. and S. Elhadad. An unsupervised aspect-sentiment model for online reviews. In *Proceedings of Annual Conference of the North American Chapter of the Association of Computational Linguistics* (*NAACL-2010*), 2010.

Dempster, P. A., Laird, M. N., and Rubin, B. D. Maximum likelihood from incomplete data via the EM algorithms. Journal of the Royal Statistical Society, Series B, 1977.

Denis, F. PAC Learning from positive statistical queries. In *Proceedings of Algorithmic Learning Theory* (*ALT-1998*), 1998.

Ding, X., B, Liu, and Philip, Yu. A holistic lexicon-based approach to opinion mining. In *Proceedings of the Conference on Web Search and Web Data Mining* (*WSDM-2008*), 2008

Domingos, P., M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. Machine Learning 29, 1997.

Downey, D., M. Broadhead and O. Etzioni. Locating complex named entities in web text. In *Proceedings of International Joint Conferences on Artificial Intelligence* (*IJCAI-2007*), 2007.

Dragut C. E., C. Yu, P. Sistla, and W. Meng. Construction of a sentimental word dictionary. In *Proceedings of ACM International Conference on Information and Knowledge Management* (*CIKM-2010*), 2010.

Esuli, A., and F. Sebastiani. "Determining term subjectivity and term orientation for opinion mining." In *Proceedings of Annual Conference of the European Chapter of the Association of Computational Linguistics* (*EACL-2006*), 2006.

Frank, E., M. Hall, and B. Pfahringer. Locally weighted Naïve Bayes. In *Proceedings of Conference on Uncertainty in Artificial Intelligence* (*UAI-2003*), 2003.

Ghahramani, Z. and K.A. Heller. Bayesian sets. In *Proceedings of Annual Neural Information Processing Systems* (*NIPS-2005*), 2005.

Girju, Roxana, Adriana. Badulescu, and Dan. Moldovan. Automatic discovery of part-whole relations. Computational Linguistics, 32(1):83-135 2006

Gliozzo, A., C. Strapparava and I. Dagan Investigating unsupervised learning for text categorization bootstrapping. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing* (*HLT/EMNLP-2005*), 2005.

Google Sets. System and methods for automatically creating lists. US Patent: US7350187, March 25, 2008.

Hall, M. A decision tree-based attribute weighting filter for Naïve Bayes. Knowledge-Based System. 2007.

Hatzivassiloglou, V., and K. McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of Joint Conference: Annual Meeting of the Association for Computational Linguistic and Conference of the European Chapter of the Association for Computational* Linguistics (*ACL-EACL-1997)*, 1997.

Hearst, M. Direction-based text interpretation as an information access refinement. Text-Based Intelligent Systems. Lawrence Erlbaum Associates, 1992.

Hu, M., and B. Liu. Mining and summarizing customer reviews. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD-2004*), 2004.

Jakob, N. and I. Gurevych. Extracting opinion targets in a single and cross-domain setting with conditional random fields. In *Proceedings of Conference on Empirical Methods in Natural Language Processing* (*EMNLP-2010*), 2010.

Jain, A. and R. Dubes. Algorithms for clustering data. Prentice-Hall, Englewood Cliffs, NJ. 1988.

Jin, W. and H. Ho. A novel lexicalized HMM-based learning framework for web opinion mining. In *Proceedings of International Conference on Machine Learning* (*ICML-2009*), 2009a.

Jin, W. and H. Ho. OpinionMiner: a novel machine learning system for web opinion mining and extraction. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD-2009*), 2009b.

Jo, Y. and A. Oh. Aspect and sentiment unification model for online review analysis. In *Proceedings of the Conference on Web Search and Web Data Mining* (*WSDM-2011*), 2011.

Joachims, T. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of European Conference on Machine Learning* (*ECML-1998*), 1998.

Joachims, T. Transductive Inference for text classification using support vector machines. In *Proceedings of International Conference on Machine Learning* (*ICML-1999*), 1999.

Kim, S., K. Han, H. Rim and S-H. Myaeng. Some effective techniques for Naïve Bayes text classification. IEEE Transaction on Knowledge and Data Engineering, vol 18, no 11. 2006.

Kleinberg, J. Authoritative sources in hyper-linked environment Journal of the ACM 46 (5): 604-632 1999.

Ko, Y. and J. Seo Learning with unlabeled data for text categorization using bootstrapping and feature projection techniques. In *Proceedings of Annual Meeting of the Association for Computational Linguistic* (*ACL-2004*), 2004.

Lafferty, J., A. McCallum, and F. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of International Conference on Machine Learning* (*ICML-2001*), 2001.

Lee, L. Measures of distributional similarity. In *Proceedings of Annual Meeting of the Association for Computational Linguistic* (*ACL-1999*), 1999.

Lee, W-S. and B. Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *Proceedings of International Conference on Machine Learning* (*ICML-2003*), 2003.

Lewis, D. and W. Gale A sequential algorithm for training text classifiers. In *Proceedings of ACM SIGIR International Conference on Information Retrieval* (*SIGIR-1994*), 1994.

Li, F., C. Han, M. Huang, X. Zhu, Y. Xia, S. Zhang and H. Yu. Structure-aware review mining and summarization. In *Proceedings of International Conference on Computational Linguistics* (*COLING-2010*), 2010.

Li, X., and B. Liu. Learning to classify texts using positive and unlabeled data. In *Proceedings of International Joint Conferences on Artificial Intelligence* (*IJCAI-2003*), 2003.

Li, X., B. Liu, S. Ng. Learning to identify unexpected instances in the test set. In *Proceedings of International Joint Conferences on Artificial Intelligence* (*IJCAI-2007*), 2007.

.

Lidstone, J. G. Note on the general case of the Bayes-Laplace formula for inductive or a posteriori probabilities. Transactions of the Faculty of Actuaries 1920

Lin, C. and Y. He. Joint sentiment/topic model for sentiment analysis. In *Proceedings of ACM International Conference on Information and Knowledge Management* (*CIKM-2009*), 2009.

Lin, D. Automatic retrieval and clustering of similar words. In *Proceedings of* In *Proceedings of Joint Conference*: *International Conference on Computational Linguistics and Annual Meeting of the Association for Computational Linguistic* (*COLING/ACL-1998*), 1998.

Liu, B. Sentiment analysis and subjectivity. A chapter in Handbook of Natural Language Processing, second edition. 2010

Liu, B., W-S. Lee, P. S. Yu and X. Li. Partially supervised text classification. In *Proceedings of International Conference on Machine Learning* (*ICML-2002*), 2002.

Liu, B., X. Li,, W-S. Lee, and P. Yu. Text classification by labeling words. In *Proceedings of National Conference of Artificial Intelligence* (*AAAI-2004*), 2004.

Liu, B., M. Hu, and J. Cheng. Opinion Observer: analyzing and comparing opinions on the Web. In *Proceedings of International Conference on World Wide Web* (*WWW-2005*), 2005.

McCallum, A. and K. Nigam. A comparison of event models for Naïve Bayes text classification. In *Proceedings of* W*orkshop on Learning for Text Categorization in National Conference of Artificial Intelligence* (*AAAI-1998*), 1998.

Mei Q., X. Ling, M. Wondra, H. Su, and C. Zhai. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of International Conference on World Wide Web* (*WWW-2007*), 2007

Neter, J., W. Wasserman, and G. A, Whitmore. Applied Statistics. Allyn and Bacon 1993.

Ng, A. and M. Jordan. On discriminative vs. generative classifiers: a comparison of Logistic Regression and Naïve Bayes. In *Proceedings of Annual Neural Information Processing Systems* (*NIPS-2002*), 2002.

Nigam, K., A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. Machine Learning 2000

Pang, B., L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*(*EMNLP-2002*), 2002.

Pantel, P. and D. Lin. Discovering word senses from text. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD-2002*), 2002.

Pantel, P., E. Crestan, A. Borkovsky, A-M. Popescu and V. Vyas. Web-Scale distributional similarity and entity set expansion. In *Proceedings of Conference on Empirical Methods in Natural Language Processing* (*EMNLP-2009*), 2009.

Popescu, A-M., and O. Etzioni. Extracting product features and opinions from reviews. In *Proceedings of Conference on Empirical Methods in Natural Language Processing* (*EMNLP-2005*), 2005.

Qiu, G., B. Liu, J. Bu, and C. Chen. Opinion word expansion and target extraction through double propagation. Computational Linguistics, 2011.

Rennie, J., L. Shih, J. Teevan, and D. Karger. Tackling the poor assumptions of Naïve Byes text classifier. In *Proceedings of International Conference on Machine Learning* (*ICML-2003*), 2003.

Riloff, E., and J. Wiebe. Learning extraction patterns for subjective expressions. In *proceedings of of Conference on Empirical Methods in Natural Language Processing* (*EMNLP-2003*), 2003.

Russell, S. and Peter. Norvig. Artificial intelligence: a modem approach. Prentice-Hall, Eng-lewood Cliffs, NJ. 2003.

Sarawagi, S. Information Extraction. Foundations and Trends in Databases 1(3): 261-377 (Foundations and Trends in Databases 1(3): 261-377), 2008.

Sebastiani, F. Machine learning in automated text categorization. ACM Computing Surveys, 34(1):1–47. 2002.

Scaffidi, C., K. Bierhoff, E. Chang, M. Felker, H. Ng, and C. Jin. Red opal: product-feature scoring from reviews. In *Proceedings of the 9th International Conference on Electronic Commerce*, 2007.

Su, Qi, Xinying. Xu, Honglei. Zhili Guo, Xian. Wu, Xiaoxun. Zhang, Bin. Swen, and Zhong. Su. Hidden sentiment association in Chinese web opinion mining. In *Proceedings of International Conference on World Wide Web* (*WWW-2008*), 2008.

Titov, I. and R. McDonald. Modeling online reviews with multi-grain topic models. In *Proceedings of International Conference on World Wide Web* (*WWW-2008*), 2008.

Turney, P. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews." In *Proceedings of Annual Meeting of the Association for Computational Linguistics* (*ACL-2002*), 2002.

Wang, R.C. and Cohen, W. W. Iterative set expansion of named entities using the web. In *Proceedings of IEEE International Conference on Data Mining* (*ICDM-2008*), 2008.

Wiebe, J. and E. Riloff. Creating subjective and objective sentence classifiers from unannotated texts. Computational Linguistics and Intelligent Text Processing, 2005: p. 486-497.

Wiebe, J., T. Wilson, R. Bruce, M. Bell, and M. Martin. Learning subjective language. Computational Linguistics, 2004, 30(3): p. 277-308.

Wilson, T., J. Wiebe, and P. Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing* (*HLT/EMNLP-2005*), 2005.

Yu, H., J. Han, K. Chang. PEBL: Positive example based learning for Web page classification using SVM. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD-2002*), 2002.

Zhang, H. and S. Sheng. Learning weighted Naïve Bayes with accurate ranking. In *Proceedings of IEEE International Conference on Data Mining* (*ICDM-2004*), 2004.

Zhao, W., J. Jiang, H. Yan, and X. Li. Jointly modeling aspects and opinions with a MaxEnt-LDA hybrid. In *Proceedings of Conference on Empirical Methods in Natural Language Processing* (*EMNLP-2010*), 2010.

Zhuang, L., F. Jing, X.-Yan Zhu, and L. Zhang. Movie review mining and summarization. In *Proceedings of ACM International Conference on Information and Knowledge Management* (*CIKM-2006*), 2006.

Zagibalov, T. and J, Carroll. Unsupervised classification of sentiment and objectivity in Chinese text. In *Proceedings of the International Joint Conference on Natural Language Processing* (*IJCNLP-2008*), 2008.

VITA

Name: Lei Zhang

Education:
M.S. Computer Science, Wuhan University, 2005
B.S. Computer Science, Wuhan University, 2002

Recent Awards:
SIGWEB Travel Award, 2011
Presenter's Travel Award, University of Illinois at Chicago, 2010, 2011
Graduate Student Council Award, University of Illinois at Chicago, 2010, 2011

Recent Publications:
Bing Liu, Lei Zhang. "A Survey of Opinion Mining and Sentiment Analysis". Book chapter in Mining Text Data : 415-463, Kluwer Academic Publishers 2012.

Lei Zhang, Bing Liu. "Extracting Resource Terms for Sentiment Analysis". In *Proceeding of the 5th International Joint Conference on Natural Language Processing* (*IJCNLP 2011*): 1171-1179.

Lei Zhang, Bing Liu. "Identifying Noun Product Features that Imply Opinions". In *Proceeding of the 49th Annual Meeting of the Association for Computational Linguistics* (*ACL 2011*): 575-580.

Zhongwu Zhai, Bing Liu, Lei Zhang, Hua Xu, Peifa Jia. "Identifying Evaluative Sentences in Online Discussions" In *Proceedings of 25th National Conference on Artificial Intelligence* (*AAAI 2011*): 933-938.

Malu Castellanos, Umeshwar Dayal, Meichun Hsu, Riddhiman Ghosh, Mohamed Dekhil, Yue Lu, Lei Zhang, Mark Schreiman. "LCI : A Social Channel Analysis Platform for Live Customer Intelligence" In *Proceedings of the 2011 ACM SIGMOD/PODS Conference* (*SIGMOD 2011*):1049-1058.

Lei Zhang, Bing Liu. "Entity Set Expansion in Opinion Documents". In *Proceedings of the 22nd ACM Conference on Hypertext and Hypermedia* (*HT 2011*): 281-290.

Malu Castellanos, Riddhiman Ghosh, Yue Lu, Lei Zhang, Perla Ruiz, Mohamed Dekhil, Umeshwar Dayal,Meichun Hsu. "LivePulse: Tapping Social Media for Sentiments in Real-Time", In *Proceedings of the 20th World Wide Web Conference* (*WWW 2011*): 193-196.

Lei Zhang, Bing Liu, Suk Hwan Lim, Eamonn O'Brien-Strain. "Extracting and Ranking Product Features in Opinion Documents", In *Proceedings of the 23rd International Conference on Computational Linguistics* (*COLING 2010*): 757-765.

Xiaoli Li, Lei Zhang, Bing Liu, See-Kiong Ng. "Distributional Similarity vs. PU Learning for Entity Set Expansion", In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (*ACL 2010*): 359-364.

Xiaowen Ding, Bing Liu, Lei Zhang. "Entity Discovery and Assignment for Opinion Mining Applications", In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD 2009*): 1125-1134.

Lei Zhang, Bing Liu, Jeffrey Benkler, Chi Zhou. "Finding Actionable Knowledge via Automated Comparison", In *Proceedings of International Conference on Data Engineering* (*ICDE 2009*): 1419-1430.