Extracting Feature Words from Customer Reviews

by

Ting Zhang

Bachelor of Software Engineering, SEU, 2012

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Computer Science

In the Graduate Academic Unit of your GAU

Supervisor(s):	Huajie(Harry) Zhang, Ph.D., Computer Science
Examining Board:	Wei Song, Ph.D., Computer Science, Chair
	Yunli Wang, Ph.D., Computer Science
	Guohua Yan, Ph.D., Mathematics and Statistics

This thesis is accepted

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

May, 2016

Abstract

Potential customers often browse online reviews before buying products. Manufacturers also collect customer feedback from the reviews. It is very hard for customers and manufacturers to get useful information from a large number of comments quickly. Thus, automatic information extraction in reviews has become a significant problem. This thesis investigates feature word extraction. Feature words are product components or attributes indicating customer interests. Since there is no systematic study on feature word extraction, we first study three classic methods: (1) the frequency-based extraction method; (2) the Web PMI-based extraction method; (3) the rapid automatic keyword extraction (RAKE) method. To provide an objective evaluation, the performance of each method is validated and compared from the following aspects: precision and recall, time complexity, and robustness. Then a new approach is proposed, the rapid feature word extraction (RFWE) method, to improve the performance. RFWE combines the techniques used in the popular methods and performs well in precision, recall, and runtime. RFWE is a great option for users to extract feature words from customer reviews.

Dedication

This thesis is dedicated to my father and mother,

Shanhong Zhang and Jianmei Zhu,

for their unconditional love and support.

Acknowledgments

First I would like to express my gratitude to my supervisor Dr. Huajie(Harry) Zhang for the useful comments and encouragement through my Master study and related research. He is really a nice and helpful person. I could not have imagined having a better supervisor for my work.

My sincere appreciation also goes to my parents, who provided me with this opportunity to study abroad. They always believe in me and encourage me to do my best. I will be grateful forever for their love.

Last but not least, I would like to thank all my friends in China and in Canada for their care and support. In particular, I am grateful to Li Ji, Fan Liang, and Lingchen Chou for all the fun we have had in the past time.

Table of Contents

List of Tables

List of Figures

List of Symbols, Nomenclature or Abbreviations

PMI	Point-wise-Mutual Information
RAKE	Rapid Automatic Keyword Extraction
RFWE	Rapid Feature Word Extraction
KEA	Keyphrase Extraction Algorithm

Chapter 1

Introduction

With the further development of the Internet, more and more people buy products and share their reviews on e-commerce sites, such as Amazon and eBay. The number of reviews online has increased over the years. Figure 1.1 shows the number of new Amazon reviews in books, which increases year after year. The data in Figure 1.1 comes from McAuley and Leskovec, who collected 34,686,770 Amazon reviews from January 1995 to March 2013 [1].

Customers usually express their own opinions and emotions on products in their reviews. From customer reviews, we can easily find the pros and cons of the products. The useful information hidden in reviews helps potential customers decide whether or not to buy the products. People have become accustomed to browsing the reviews of other customers before buying a prod-

uct. On the other hand, manufacturers also want to collect meaningful feedback from customers, so that they can know better about customer needs and preferences. Mining customer reviews provide manufacturers with a great way to find all available future directions of their products and attract more customers.

Figure 1.1: New Amazon Reviews in Books [1]

However, some reviews are very long and even contain irrelevant things about products. It is very hard for customers and manufacturers to get useful information from a large number of comments quickly. As a result, automatic information extraction in product reviews has become a significant problem. This extraction can be described as a text mining problem, which helps users

to make sense of customer reviews and generate useful summaries.

According to the automatically extracted feature words, we can easily know customer focuses and concerns. It has a practical significance to potential customers and manufacturers. On the other hand, as an important part in sentiment analysis research, feature word extraction also has high research value. This thesis describes the problem of feature word extraction from the above two aspects.

1.1 Feature Word Extraction

This thesis mainly investigates the problem of feature word extraction. The purpose of feature word extraction is to identify feature words in customer reviews through approaches such as lexical analysis, association mining, and pattern recognition.

Feature words can be product components or attributes on which opinions are expressed. They are usually nouns and noun phrases in reviews. However, due to the intricacy of linguistics, implicit features are difficult to handle in certain contexts [2]. Let us see an example sentence from customer reviews of a camera:

"It was too expensive for the shots that I got."

This example sentence tells us that the user is disappointed with the price of the camera and "price" is the product feature that the user is talking about. But the word "price" does not appear in the sentence and is obtained from word "expensive". The human brain can understand the hidden meaning, but it is hard for a computer to deal with a sentence like this. In this thesis, we focus on explicit feature words that appear in the reviews, like nouns and noun phrases.

1.2 Feature Extraction in Sentiment Analysis

Feature word extraction is an important task in sentiment analysis. As a popular research field in text mining, sentiment analysis has attracted a lot of attention from researchers. This problem covers techniques and studies for natural language analysis and text mining [3]. In general, sentiment analysis can be divided into two primary tasks [4]: feature word extraction and opinion phrases identification.

Feature word extraction: In general, feature words indicate product attributes and functions [2]. Given a collection of customer reviews of a single product (e.g. Canon camera), this process extracts product features (e.g. picture quality or battery) based on the association between feature words and the product. Feature word extraction is also known as keywords, topic, aspect, or opinion target extraction [4]. This task is the main topic of my

thesis.

Opinion words extraction: This process automatically classifies the polarity of opinion words for each product feature in reviews. Opinion words indicate positive or negative sentiments, such as *good*, *bad*, *amazing* and *terrible*. For example, in a camera review, *"its battery life is absolutely amazing!"*, "battery life" is a product feature and "amazing" indicates a positive opinion. From this review, people can easily know that this customer is satisfied with the battery life of this camera.

Thus, if potential customers are interested in a specific product feature, they can easily find out whether other customers are satisfied with it through sentiment analysis. It also provides an efficient way for manufacturers to collect product information and analysis their competitors.

1.3 Objectives

This thesis focuses on feature word extraction, which has great value in research and applications. Feature words indicate customer concerns, which are of great value to potential customers and manufacturers. The research of feature word extraction has a very practical significance.

The study on feature word extraction is far from enough now. Although several methods have been proposed to solve the problem, there is no sys-

tematic study on feature word extraction. To make up the research gap, we conduct this research. On the other hand, to improve the performance of feature word extraction, this thesis propose a new approach based on the research. We hope that this thesis can provide guidelines for future research and applications.

In order to make further progress in feature word extraction, current approaches need to be investigated first. Our plan is to choose three popular feature word extraction methods and systematically study these methods by experiments. Through experimental study, we expect to find out the pros and cons of each method. Then, we propose improvements and explore a new approach based on the experiments on these popular methods.

In this thesis, we choose three methods widely used in research. They are:

- The frequency-based extraction method
- The Web Point-wise-Mutual Information (PMI) based extraction method
- The rapid automatic keyword extraction (RAKE) method

The reason we choose these three methods is that they all represent some of the common feature word extraction methods and can give us a general understanding of this problem. The Frequency-based method is a simple statistical way to extract feature words. The Web PMI-based method is a

syntactical way that uses discriminators to find candidates. The discriminators are simple patterns that are associated with the product.

Unlike the above methods, the RAKE method is a keyword extraction method, which focuses on the operating speed. The reason we choose the RAKE method is that keyword extraction is similar to feature word extraction. They both extract information from the text. The very difference is that keyword extraction is used to extract the terms that represent the subject of a document. These terms help users understand the main idea of the document quickly. While feature word extraction is to extract product features in reviews which are only related to the products. There is a subtle difference here between the main idea of an article and the product features.

Feature word extraction usually includes three steps: (1) identifying candidate feature words (candidates, for short); (2) evaluating candidates based on certain criteria; (3) removing noisy words by pruning.

The first method is a well-known frequency-based mining method proposed by Hu and Liu [2]. It returns high-frequency nouns and noun phrases in reviews. This popular method also includes a pruning mechanism to remove candidates that are unlikely to be product features. Hu and Liu mentioned two types of pruning in their work: compactness pruning and redundancy pruning.

The second method adds an extra pruning step to Hu and Liu's method. The

new pruning step applies the conception of PMI [5], which is used to evaluate the association between candidates and some predefined discriminators. The PMI score indicates the possibility that one word collocates with another word. This method also queries the Web as a corpus to compute PMI scores, which costs time. And we call it the Web PMI-based extraction method.

The last but not least, the RAKE method extracts keywords that contain multiple words except punctuation marks and stop words [6]. Unlike the other two methods, the RAKE method is not corpus-oriented and extracts candidates by computing word co-occurrence scores. The co-occurrence score is meaningful and indicates the possibility that several words appear alongside each other in a document. Furthermore, the RAKE method is designed without natural language processing, which makes it more efficient than other methods.

As we know, various methods have their own distinct characteristics and limitations. We plan to compare the performance of each method in various measures, such as precision, recall, time complexity, and robustness. Our goal is to provide an objective evaluation of these popular methods and to find out the situations in which they can work properly. After we systematically know about each method, we propose a new approach to improve the performance. As a combination of these popular methods, the new approach holds the advantages over all the methods. The performance of the new approach is validated through comparative analysis.

This thesis is organized as follows: Chapter 2 describes background and literature review. Chapter 3 presents the details of the three feature word extraction methods. Chapter 4 gives an objective experimental evaluation. Chapter 5 gives the idea and implementation of the new method. Chapter 6 presents the conclusion.

Chapter 2

Background and Literature Review

This chapter is to illustrate the relevant studies in feature word extraction. As we discussed before, the goal of this thesis is to provide a systematic study in feature word extraction and propose some improvements on the existing methods. To achieve this goal, we investigate current approaches and widely used techniques in this field.

As a branch of text mining, feature word extraction has great value both in research and applications. To get a big picture and a better understanding of this issue, we study text mining first. Many concepts and techniques in text mining are inspiring and can also be applied in feature word extraction.

2.1 Text Mining

Text mining is an interdisciplinary problem, which includes data mining, machine learning, natural language processing, information retrieval, and knowledge management [9]. The main task of text mining is to extract information hidden in the text by identifying patterns and trends.

However, since the data sources in text mining are text documents, it is not easy to find patterns and trends in unstructured data. Thus, text mining usually starts with a preprocessing of documents for textual data normalization, such as lexical analysis or syntactic analysis. The preprocessing of documents converts the original documents into a structured format, so that a text mining system can apply data mining methods on the documents and extract key information.

After the text normalization, statistical methods can be applied to the structured data to identify patterns and trends, which can be used to extract information that people might be interested in. This thesis is about product feature extraction in customer reviews, so the information of interest here mainly refers to product features that customers talked about in their reviews.

As we know, text mining has been a hot research topic and has a lot of applications in many fields. It involves a variety of problems, such as text

summarization, text classification, sentiment analysis, keyword extraction, etc. Text mining also has a wide range of significant applications, e.g. marketing analysis and content analysis.

In this thesis, we discuss the problem of text mining in customer reviews. Below we will introduce several approaches to sentiment analysis and keyword extraction. These approaches are also effective to extract product features in customer reviews.

2.2 Sentiment Analysis

The work of sentiment analysis is to recognize and classify emotions. The phrases, opinion mining, and subjectivity analysis, are also used. Many studies have been conducted on sentiment analysis (e.g. Yi et al., 2003 [10], Hu and Liu, 2004 [2], Zhuang et al., 2006 [12], Popescu and Etzioni, 2007 [5], Scaffidi et al., 2007 [17], Kobayashi et al., 2007 [14], Qiu et al., 2009 [15], Zhang et al., 2010 [16]). Researchers have proposed several methods to solve this problem.

As one of the primary tasks of sentiment analysis, most existing feature word extraction methods identify statistically discriminating words across a corpus based on frequency or relationships between words. For example, Yi et al. (2003) introduced a likelihood-ratio test to identify feature terms in online

reviews [10]. A likelihood score implies a confidence level that a candidate feature is relevant to a product. The higher the likelihood value is, the more relevant the candidate feature is to the product.

Hu and Liu (2004) proposed a mining method based on word frequency in their system [2], which applies association mining [11] to find high-frequency words. The main idea of this algorithm is that customers prefer to use same words to describe product features when they write reviews on a product. Thus, high-frequency nouns and noun phrases in reviews are more likely to be product features. A pruning mechanism is included to remove meaningless candidates. The system also extracts opinion words. The resulting opinion words are used to find relevant infrequent features.

In the work of Popescu and Etzioni (2007), the authors proposed OPINE, an information extraction system, to extract features and associated opinion words [5]. Unlike other methods, the OPINE system utilizes the Web as a corpus to determine whether a noun or noun phrase is a feature. It queries the Web to get the hit counts of words, which takes a long time. The OPINE system calculates the PMI value between candidates and automatically generated discriminators (e.g., "[P] has", "is a [P]", where [P] is a product type and needs to be known in advance).

Following the idea that opinion words can be used to identify features, several methods have been proposed. Zhuang et al. (2006) introduced a **depen-**

dency grammar graph to mine the relations between features and opinion words in movie reviews [12]. The dependency grammar graph is based on the idea of universal dependencies which describe the grammatical relations in a sentence [13]. The resulting relations or patterns help extract feature-opinion pairs in other sentences.

For instance, in the sentence *"The camera is an excellent choice"*, we can easily get the shortest path between "camera" and "choice" by parsing: "*camera(NN) - nsubj - is(VBZ) - cop - choice(NN)*". The frequent patterns can be used to extract feature-opinion pairs in other similar sentences.

Kobayashi et al. (2007) proposed an approach which combines contextual and statistical clues [14]. Here contextual clues are syntactic patterns which can be used to extract relations between opinions and features. Statistical clues refer to the scores of feature-feature and feature-opinion co-occurrence.

Qiu et al. (2009) proposed a **double propagation** approach, which generates rules based on dependency relations between features and opinion words, and also features and opinion words themselves [15]. In the work of Zhang et al. (2010) [16], "part-whole" patterns and "no" patterns are used to improve **double propagation**. Their system ranks the candidates by feature importance, which is calculated based on feature relevance and feature frequency.

2.3 Keyword Extraction

Keyword extraction is also a hot research field in text mining. Keywords are usually used to express the primary information of a document [19]. The applications of keyword extraction have extended into many fields, such as information retrieval (IR) systems [20], topic mining, and news summarization.

In keyword extraction, there are two types of approaches: corpus-oriented and document-oriented [6]. Early methods proposed to extract keywords focus on corpus-oriented research, which evaluates statistics of individual words in a set of documents. However, corpus-oriented methods can miss candidates which only appear in some documents. These candidates are not statistically identifiable in corpus. Besides, corpus-oriented methods prefer to extract single words. To avoid these problems, document-oriented methods are designed to operate on a single document. These methods extract context-independent features from a document [19].

Ian et al. (1999) [21] proposed an automatic keyphrase extraction algorithm (KEA). KEA identifies candidates through lexical analysis and extracts keywords by training documents. A prediction model is built and used to find more keywords in new documents.

Hulth (2004) [22] combined natural language processing and machine learn-

ing, and compared the effectiveness of three term identification methods: noun phrase chunks, n-grams, and POS tags.

Rose et al. (2012) proposed the rapid automatic keyword extraction (RAKE) method, which computes word co-occurrence within candidates [6]. The RAKE method is a document-oriented method and extracts keywords that contain multiple words. Moreover, the RAKE system is designed without natural language processing, which makes it much quicker.

Chapter 3

Feature Word Extraction

In our feature word extraction study, we first implement three classic extraction algorithms: (1) the frequency-based extraction method; (2) the Web PMI-based extraction method; (3) the rapid automatic keyword extraction (RAKE) method. These algorithms have their own strengths and weaknesses. Our work is to evaluate these methods by experiments.

The first two methods both find candidates based on word frequency in documents. The difference between them is that the Web PMI-based extraction method utilizes the Web as a corpus to prune candidates. The frequency-based extraction method only uses the local documents. These two methods both focus on evaluating statistics of individual words at corpus level.

However, as discussed before, corpus-oriented methods can miss important

feature words that cannot be statistically identified in the corpus. Besides, corpus-oriented methods usually tend to focus on single words. According to [6], the RAKE method is designed to avoid the drawbacks of corpus-oriented methods. It focuses on word co-occurrence and extracts keywords which contain multiple words. Furthermore, as its name suggests, the RAKE method is rapid and efficient. It does not rely much on natural language processing that costs a lot of time.

3.1 Overview

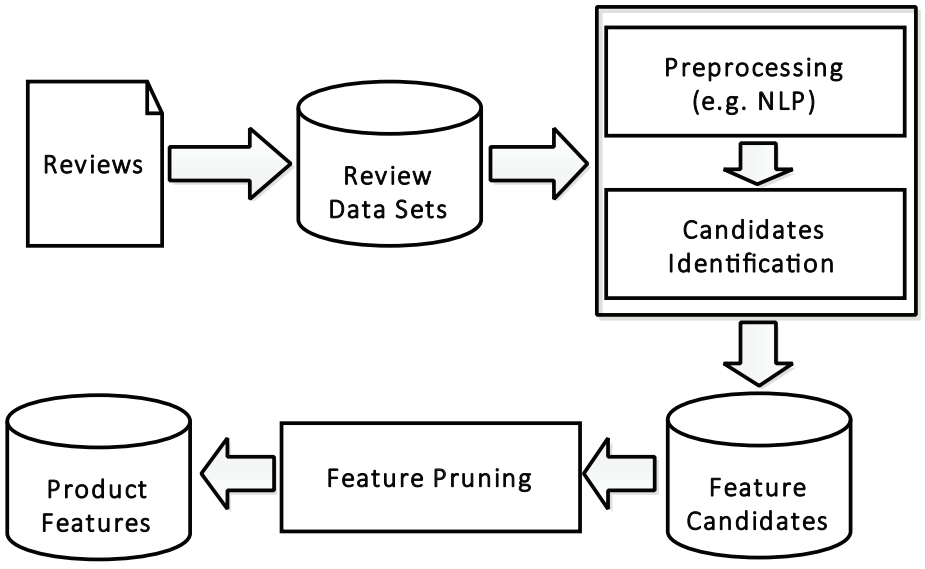

Figure 3.1: Feature word extraction

As discussed before, this thesis implements three feature extraction methods respectively. All of these methods perform feature word extraction in three

main steps: (1) finding candidate feature words in each review; (2) evaluating the correlation between product and candidate feature words; (3) purifying the results by pruning. Figure 3.1 shows the process overview of feature word extraction, which is a universal solution to this problem.

Given the inputs, the program first reads in all reviews of one product. After that, it identifies candidate feature words that meet certain standards. Note that natural language processing, like parser and part-of-speech (POS) tagging, helps us to find candidate feature words. In the last step, pruning methods are applied to filter candidates that are less relevant to the product. In each process, different methods may have their own implementation techniques. Below, we describe each method separately.

3.2 Frequency-based Extraction

As we discussed before, the well-known frequency-based extraction method, proposed by Hu and Liu (2004), considers all nouns and noun phrases in reviews as candidates. In this thesis, we use Stanford CoreNLP as a suite of natural language processing tools to find all candidates from reviews.

3.2.1 Stanford CoreNLP

Stanford CoreNLP [23] is an integration of natural language analysis tools, including POS tagger, the parser, etc. For each input text, Stanford CoreNLP generates XML output with all relevant annotation, like POS tag, lemma, etc. Here we use Stanford CoreNLP to parse reviews and split sentences into words, in such way that the POS tag for each word can be obtained. With a POS tag, we know whether a word is a noun, verb, adverb, etc. Moreover, we can also find noun phrases in sentences. The following shows a simple sentence from the reviews of a canon camera:

"I recently purchased the canon powershot g3 and am extremely satisfied with the purchase."

After parsing the sentence using Stanford CoreNLP, we get a parse tree with the POS tags. Here "NN" refers to a noun and "NP" refers to a noun phrase.

```
(ROOT (S (NP (PRP I)) (ADVP (RB recently)) (VP (VP (VBD
purchased) (NP (DT the) (NN canon) (NN powershot) (NN g3)))
(CC and) (VP (VBP am) (ADJP (RB extremely) (VBN satisfied)
(PP (IN with) (NP (DT the) (NN purchase)))))) (. .)))
```

We can easily find the nouns and noun phrases from this output generated by Stanford CoreNLP. Words with other POS tags in the sentence are not likely to be features, such as "recently", "satisfied", etc.

Some other pre-processing can also be performed during this process, such as removing stop words and lemmatization. Stop words usually refer to common words. For instance, we have two noun phrases in the sentence above, "the canon powershot g3" and "the purchase". It is obvious that word "the" is not a product feature and belongs to stop words that we do not care about. Lemmatization is used to convert words into their base forms. In general, words appear in various forms with prefixes, suffixes or infixes. For example, word "produce" has other forms , such as "produces", "produced", "producing". The base word "produce" is called the lemma. It is easier to use lemmas to count word frequencies.

3.2.2 Pruning

Pruning is used to remove candidates that are not likely to be product features. Hu and Liu mentioned two types of pruning in their work: compactness pruning and redundancy pruning.

Compactness pruning mainly focuses on candidates that contain more than one word. The goal of compactness pruning is to remove multi-words candidates in which words do not appear together.

In a sentence, words are more likely to be meaningful if they appear together in a certain order. Take a multi-word candidate c as an example. If c is not compact in a sentence, it means that there are more than three other

words between any two adjacent words of c. For example, candidate "canon camera" is not compact in the sentence "canon has brought a revolution to digital camera technology".

On the other hand, if c is compact in more than one sentence, c can be considered as a compact phrase. If we cannot find two or more sentences that contain c as a compact phase, we discard c from candidates.

Redundancy pruning is to remove redundant candidates. Some candidates are subsets of other candidates. For example, "battery" is a subset of "battery life". Thus, it could be confusing when counting word frequencies. To avoid this problem, we use *p-support* to evaluate candidates.

The *p-support* of a candidate c indicates the number of sentences that contain c. Meanwhile these sentences do not contain another candidate that is a superset of c. For example, we have sentence "the battery life is wonderful". It can be counted in the *p-support* of "battery life", but not "battery". Besides, we set a minimum *p-support* to prune redundant candidates. If the *p-support* of a candidate c is lower to the minimum *p-support* and c is a subset of another candidate, c is removed.

3.2.3 The Frequency-based Extraction Algorithm

In this part, we focus on the algorithm of the frequency-based extraction method. First, let us introduce a basis data type *PhraseInfo*. The idea of data type *PhraseInfo* is to store the resulting candidates, including the information we need, like lemma string, weight, etc. Since there are three feature word extraction algorithms, we use "weight" to unify names. Different algorithms have their own way to calculate weight. Here "weight" presents word frequency.

The complete procedure of the frequency-based extraction algorithm is shown as Algorithm 1. The input is a list of customer reviews in which all words are converted to lowercase. The goal of this method is to return a list of features ordered by their weights.

The method first extracts all distinct nouns and noun phrases (lines 4-6 in Algorithm 1) and stores them as candidates. Then it calculates the *p-support* (lines 7-13) and compact degree (lines 15-21) of each candidate, as discussed before. After that, the method prunes wrong candidates to purify results (lines 22-26). If the compact degree of candidate c is smaller than two, which means that c is not compact in most sentences, c is removed. Besides, if the *p-support* of candidate c is smaller than three and c is a subset of another candidate *c'*, c is pruned. After the pruning process, the remaining candidates are returned in descending order of the weights (*p-support*).

Algorithm 1 Frequency-based Extraction [2]

Require: *reviewslist*: a list of reviews
Ensure: *candidates*: a list of candidates after pruning

```
1: function FrequencyExtraction(reviewslist)
2:     candidates = new List(PhraseInfo)
3:     psupport = new Map(PhraseInfo, int)
4:     for sentence in reviewslist do
5:         candidates.add(Parser.GetNounsAndPhrases(sentence))
6:     end for
7:     for sentence in reviewslist do
8:         for c in candidates and c in sentence do
9:             if ∄ c': c' in candidates and c' in sentence and c in c' then
10:                psupport[c']++
11:            end if
12:        end for
13:    end for
14:    compact = new Map(PhraseInfo, int)
15:    for c in candidates do
16:        for sentence in reviewslist do
17:            if MaxWordsDistance(c, sentence) < 3 then
18:                compact[c]++
19:            end if
20:        end for
21:    end for
22:    for c in candidates do
23:        if (psupport[c] < 3 and ∃ c': c' in candidates and c in c') or
   compact[c] < 2 then
24:            candidates.remove(c)
25:        end if
26:    end for
27:    for c in candidates and psupport[c]> threshold do
28:        c.Weight = psupport[c]
29:    end for
30:    candidates.SortByWeight()
31:    return candidates
32: end function
```

3.3 Web PMI-based Extraction

We extend the frequency-based extraction by including an additional pruning process that evaluates candidates based on the PMI score (Popescu and Etzioni, 2007). We call it as Web PMI-based extraction method. It uses the Web as a corpus to measure the PMI score between each candidate and some predefined discriminator phrases. These predefined phrases are associated with the product type.

3.3.1 Point-wise-Mutual Information (PMI)

The PMI score is widely used in statistics to measure the association between two words in the same context. It indicates the possibility that one word collocates with another word. This is why the PMI score can be used to evaluate the association between a candidate and a product. Let us first see the definition of the PMI score.

Definition 3.3.1. *C is a set of candidates and D is a set of discriminator phrases of a product type. Given $c \in C$, $d \in D$, the PMI score of c and d is defined as follows:*

$$PMI(c, d) = \frac{Hits(c + d)}{Hits(c) * Hits(d)};$$

Discriminator phrases are simple extraction patterns that are associated with the product type. They can be used to validate candidates. For example, "of digital camera" is a discriminator phrase for "digital camera". Candidates and discriminator phrases are combined as search queries. The hit counts of these queries are returned from Google search engine and stored into a local database as a resource, so that we do not need to query the Web every time. Here we use MySQL as the local database to store web hits count. Table 3.1 shows the discriminator phrases that are used to estimate features of a product [24].

C of (*) P	P contain (*) C	C for (*) P
P with (*) C	P has (*) C	C (*) in (*) P
P come with (*) C	P equipped with (*) C	P endowed with C

Note: P is product type and C refers to candidate features.

Table 3.1: Discriminator Phrases

Take digital cameras as an example. In order to estimate the possibility that candidate "battery life" is a feature of a digital camera, the method calculates the PMI score between "battery life" and the discriminator phrases, such as "of digital camera". If the PMI score is higher than the threshold, it indicates that "battery life" is a meaningful feature of product type "digital camera".

3.3.2 The Web PMI-based Algorithm

In this part, we focus on the implementation of the Web PMI-based extraction method. As we discussed before, it extends the frequency-based extraction method by including an extra pruning method. The extra pruning process is invoked after line 32 of Algorithm 1.

The complete procedure of the extra pruning method is shown as Algorithm 2. The input is a list of candidates ordered by frequencies. The method first finds the candidates with low frequencies and puts them in list *freqPruning* (lines 4). Then it calculates the PMI score of each candidate (lines 5-7). The pruning method puts all candidates with low PMI scores in list *pmiPruning* (line 8). If candidate c appears in both pruning lists, c is removed (line 9-11). Since all the hit counts of candidates are stored in the local database, function GetPMIScore directly reads hit counts from the database (lines 19-23).

Algorithm 2 Web PMI-based Algorithm [5]

Require: *candidates*: a list of candidates ordered by frequency, *discriminators*: discriminator phrases
Ensure: *candidates*: a list of candidates ordered by the Web PMI scores

```
1: function WebPMIExtraction(candidates, discriminators)
2:     freqPruning = new List(PhraseInfo)
3:     pmiPruning = new List(PhraseInfo)
4:     freqPruning = candidates.GetLowWeight()
5:     for c in candidates do
6:         c.Weight = GetPMIScore(c.lemmaString, discriminators)
7:     end for
8:     pmiPruning = candidates.GetLowWeight()
9:     for c in candidates and c in freqPruning and c in pmiPruning do
10:        candidates.remove(c)
11:    end for
12:    candidates.SortByWeight()
13:    return candidates
14: end function
15:
16: function GetPMIScore(string, discriminators)
17:    pmiScore = 0
18:    for d in discriminators do
19:        if d.isLeft then
20:            pmiScore+=DataBase.getCount(string+d)/DataBase.getCount(d)
21:        else
22:            pmiScore+=DataBase.getCount(d+string)/DataBase.getCount(d)
23:        end if
24:    end for
25:    pmiScore = pmiScore/DataBase.getCount(string)
26:    return pmiScore
27: end function
```

3.4 RAKE

The RAKE method is designed as a rapid, document-oriented method that finds keywords (or features) with multiple words (Rose et al., 2012). Here we take product features as keywords. Since natural language processing costs much time, the RAKE method parses words in a document by stop words and punctuation marks to identify candidates. Then it measures the co-occurrence score of each candidate based on the degree and frequency of words in the candidate [6].

3.4.1 Parsing

Feature words rarely contain punctuation marks and stop words, such as "and", "the", "of", etc. Thus, the RAKE method parses sentences using punctuation marks and stop words, which makes it more efficient than other methods.

Algorithm 3 gives the parsing procedure. The input is a list of reviews. The goal of this procedure is to identify the candidates in reviews. It first splits the reviews into sentences, and scans each word in each sentence (line 4-5). If a word is a stop word or is not a noun, we take the word as a separator. The words between two separators are combined as a candidate (line 6-11). Here we use WordNet [25], a lexical database, to determine the POS tags of

the words.

Algorithm 3 Parsing [6]

Require: *reviewslist*: a list of reviews
Ensure: *candidates*: a list of candidates after parsing

```
1:  function RAKEParsing(reviewslist)
2:      candidates = new List(PhraseInfo)
3:      c = new String()
4:      for sentence in reviewslist do
5:          for word in sentence do
6:              if !WordNet.isNoun(word) or word.isStopword() then
7:                  candidates.add(c)
8:                  c = new String()
9:              else
10:                 c.add(word)
11:             end if
12:         end for
13:     end for
14:     return candidates
15: end function
```

3.4.2 Co-occurrence

After the identification of candidates, the co-occurrence score of each candidate is calculated. This score is meaningful to evaluate the possibility that several words appear alongside each other in a document. The following is the definition of the co-occurrence score:

Definition 3.4.1. *C is a set of candidates. Given $c \in C$ and $w_i \in c$, the*

co-occurrence score of c is defined as follows:

$$CO(c) = \sum_{i=1}^{n} \frac{degree(w_i)}{frequency(w_i)};$$

For candidate c, its co-occurrence score is defined as the sum of the weights of all the words within c. Here the weight of word w is the ratio of *degree(w)* to *frequency(w)*. Word degree refers to the number of words that appear with w in all candidates, and word frequency is the number of w.

To better illustrate the definition of the co-occurrence, let us see a simple example. There are 10 candidates with 9 distinct words inside. The weight of each word is shown in Table 3.2.

```
battery life, image quality, color, battery, canon camera, software,
image, canon, option, software
```

	battery	life	image	quality	color	canon	camera	software	option
degree(w)	3	2	3	2	1	3	2	2	1
frequency(w)	2	1	2	1	1	2	1	2	1
degree(w)/frequency(w)	1.5	2	1.5	2	1	1.5	2	1	1

Table 3.2: Word Weight

Based on Table 3.2, we can easily get the co-occurrence score of each candidate by adding up the weight of each word inside. The following shows the candidates with their co-occurrence scores. If the co-occurrence score of a candidate is higher than the threshold, it is considered as a product feature.

```
battery life (3.5), image quality (3.5), color (1), battery (1.5),
canon camera (3.5), software (1), image (1.5), canon (1.5),
option (1), software (1)
```

3.4.3 The RAKE Algorithm

This part gives the implementation details of the RAKE method. Its key procedure is shown in Algorithm 4. The input is a list of candidates generated by parsing.

Algorithm 4 first counts the degrees and frequencies of all words in the candidates (lines 4-9). Then it computes the co-occurrence score of each candidate by adding up the weights of the words inside(lines 10-14). Finally, the candidates with high co-occurrence scores are returned as product features.

Algorithm 4 RAKE Algorithm [6]

Require: *candidates*: a list of candidates after parsing
Ensure: *candidates*: a list of candidates ordered by the co-occurrence scores

```
1:  function RAKE(candidates)
2:      frequency = new Map(String, int)
3:      degree = new Map(String, int)
4:      for c in candidates do
5:          for word in c do
6:              frequency[c]++
7:              degree[c]+=c.length
8:          end for
9:      end for
10:     for c in candidates do
11:         for word in c do
12:             c.Weight+=degree[word]/frequency[word]
13:         end for
14:     end for
15:     for c in candidates and c.weight< threshold do
16:         candidates.remove(c)
17:     end for
18:     candidates.SortByWeight()
19:     return candidates
20: end function
```

Chapter 4

Experiments and Evaluation

To implement these three feature word extraction methods and compare their effectiveness, a software workbench called Weka [26] is introduced in our research. Section 4.1 gives some details about the workbench, e.g. the data format and pre-processing. The experiment results are shown and evaluated in Section 4.2. Through experiments, the performance of each method is validated and compared from the following aspects: precision, recall, time complexity, and robustness. After all the analysis and evaluation, some specific assessments of these feature word extraction methods are provided.

4.1 Weka Overview

Weka is a collection of standard machine learning algorithms, such as linear regression, naive Bayes, random forest etc. Besides, it includes several tools for data pre-processing, classification, clustering, association, and visualization [26]. Weka also allows us to develop and compiles our own machine learning algorithms. Thus, it is well suited for developers to implement new machine learning algorithms.

As an easy-to-use open source software in Java, Weka workbench provides several applications to simplify the processes of machine learning. There are four main interfaces in Weka. Here we mainly use the Explorer interface, which has six application options: Preprocess, Classify, Cluster, Associate, Select attributes and Visualize. Each option corresponds to relevant machine learning task. For example, in Preprocess panel, we can convert from a database to ARFF (see Subsection 4.1.1) and read in data. The Explorer interface provides many standard data learning tools and algorithms. Our feature word extraction methods are also included in this interface.

4.1.1 Attribute-Relation File Format

To normalize the input data from different sources, Weka applies a data format called the Attribute-Relation File Format (ARFF). An ARFF file

usually contains a list of instances with several attributes and attribute types. The following is an example ARFF file:

```
@relation Canon-G3

@attribute title string
@attribute content string

@data
"canon","i have owned this camera for a short time and
wouldn't give it up for anything. it surpasses my greatest
expectations in a 4mp camera."
"noise level","this camera has significantly more noise at
iso 100 than the nikon 4500."
```

According to this example, it is clear that an ARFF file has two sections: the header section and the data section. The header section of an ARFF file gives the file description and data definitions. We can define the relation name, the attributes and their types. There are four data types supported in Weka: numeric, string, date and nominal. The declaration format is:

```
@relation <relation name>
@attribute <attribute name> <data type>
```

The data section of an ARFF file begins with tag `@data` and contains a set of instances with attribute values. The attribute values in each instance are separated by commas. Note that the order of the attribute declarations in

header section indicates their positions in instances.

4.1.2 Data Pre-Processing

The datasets used in our experiments are from Hu and Liu [2], including annotated customer reviews of five products from Amazon. The basic statistics of the datasets are provided in Table 4.1. All the reviews in the datasets are manually evaluated. We can compare the automatically extracted feature words with the manual features and evaluate the effectiveness.

Dataset	NO. of Reviews	NO. of Sentences	NO. of Target Feature Words
Digital Camera1	45	597	89
Digital Camera2	34	346	69
Phone	41	546	98
MP3 Player	95	1716	149
DVD Player	99	740	96

Table 4.1: The Basic Statistics of the Datasets

To extract feature words, we need to convert the datasets of product reviews into ARFF files. Since feature word extraction mainly focuses on the contents, additional information, such as username, time, and star rating, is not necessary for our experiments.

Thus, there are only two attributes declared in data files, title, and content,

like the example ARFF file in Subsection 4.1.1. The two attributes are both defined as string type. In data part, each customer review is an instance with two attribute values. The attribute values in each instance appear in the same order as their corresponding declaration.

4.2 Experimental Evaluation

After pre-processing, we conducted experiments on the five datasets. This section provides objective comments on the performance of the three feature word extraction methods. It includes precision and recall analysis, time complexity analysis, and robustness analysis. These analyses indicate the pros and cons of each method, which have great guidance value in research and applications.

4.2.1 Precision and Recall Analysis

Precision and recall are common indicators to characterize the effectiveness of the feature word extraction methods (e.g. Yi et al., 2003, Hu and Liu, 2004, Zhuang et al., 2006, Popescu and Etzioni, 2007, etc.).

First, let us see some definitions of precision and recall. In machine learning, there are some terms defined to describe the prediction results.

True positives (TP) mean the retrieved results that are true or relevant.

False positives (FP) mean the retrieved results that are not relevant.

True negatives (TN) mean irrelevant results that are not selected.

False negatives (FN) mean the relevant results that not selected.

Here **positive** and **negative** mean whether an instance is retrieved. **True** and **false** refer to whether the prediction is the same as the observation.

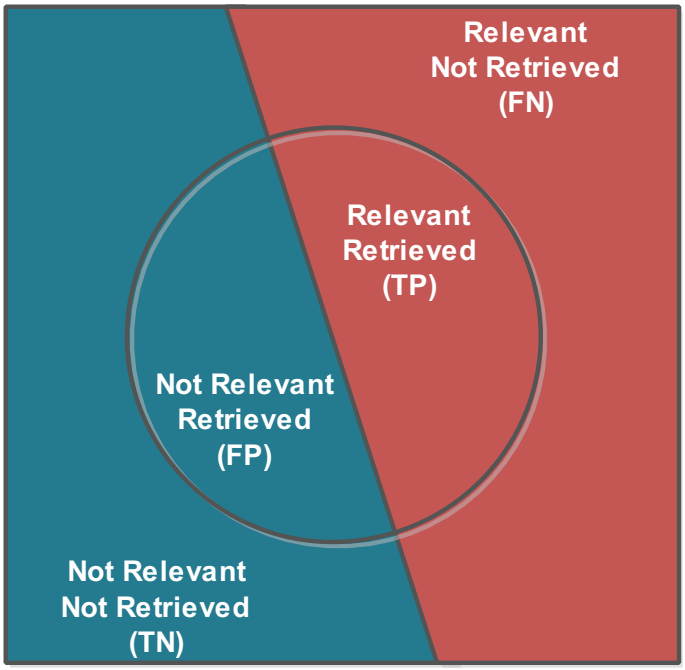

Figure 4.1: Precision and Recall

Definition 4.2.1.

$$Precision = \frac{TP}{TP + FP}; Recall = \frac{TP}{TP + FN};$$

Figure 4.1 helps us better understand the significance of precision and recall. As shown in the figure, precision is defined as the possibility of the selected part that is relevant. Recall refers to the possibility of the relevant part that is selected. In general, high precision means low recall. What we need to do is to find a balance between these two indicators.

Table 4.2 and Table 4.3 give the precision and recall results of the three feature word extraction methods. In order to visually analyze the results, we convert the tables into Figure 4.2 and Figure 4.3 respectively.

As shown in the Figure 4.2 and Figure 4.3, the precision, recall results of the RAKE method are the lowest of all and are much lower than the other two methods. As we discussed before, unlike the other two methods, the RAKE method is a keyword extraction method, which mainly extracts phrases that contain multiple words. This method is not sensitive to single words which may also be feature words. Besides, to guarantee the operating speed, the RAKE method is particularly designed as a rapid method without natural language processing, which also affects the accuracy of the algorithm. This is why RAKE method has no advantage in precision and recall analysis.

For the other two methods, the Web PMI-based method's precision is 3% higher than the frequency-based method at a cost of 10% recall drop. Since the cost is too much, the extra pruning process in the Web PMI-based method, which excludes candidates by computing their PMI scores, is not

efficient enough. Thus, it is not suitable to use the Web as a corpus for pruning in this case.

On the other hand, to better understand the precision and recall results, we introduce the F-measure score. As a combination of precision and recall, the F-measure score is the weighted harmonic mean of precision and recall.

Definition 4.2.2.

$$F - Measure = 2 * \frac{Precision * Recall}{Precision + Recall};$$

The F-measure results are provided in Table 4.4, which are calculated based on the precision and recall results in Table 4.2 and 4.3. As shown in Figure 4.4, the frequency-based method's F-measure score is 6% higher than the Web PMI-based method and is 22% higher than the RAKE method. It is clear that the frequency-based method works better than the other two methods. The RAKE method doesn't perform well in this measurement.

Dataset	NO. of Manual Features	Frequency-based	Web PMI-based	RAKE
Digital Camera1	89	0.78	0.82	0.63
Digital Camera2	69	0.88	0.91	0.4
Phone	98	0.76	0.81	0.51
MP3 Player	149	0.68	0.70	0.62
DVD Player	96	0.83	0.83	0.81
Average	100	0.78	0.81	0.60

Table 4.2: Precision

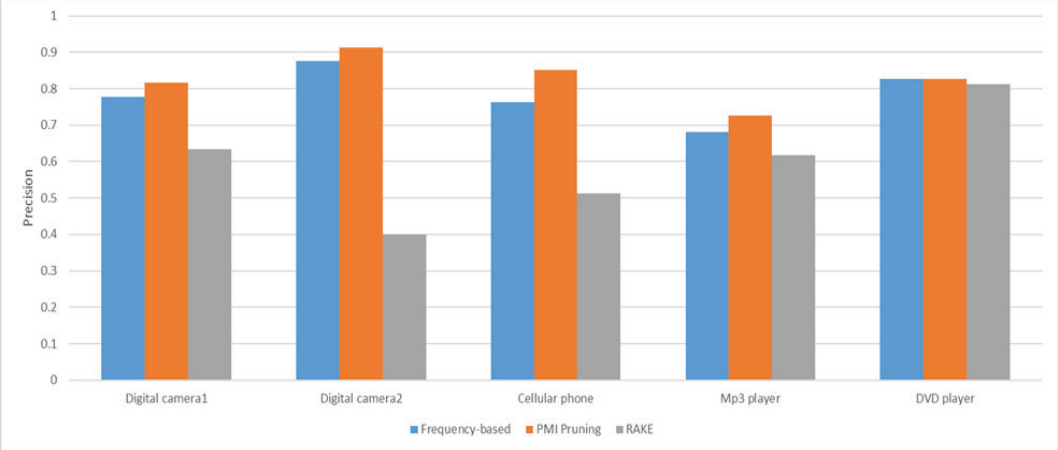

Figure 4.2: Precision

Dataset	NO. of Manual Features	Frequency-based	Web PMI-based	RAKE
Digital Camera1	89	0.73	0.62	0.45
Digital Camera2	69	0.61	0.57	0.32
Phone	98	0.64	0.59	0.40
MP3 Player	149	0.64	0.48	0.41
DVD Player	96	0.60	0.46	0.45
Average	100	0.64	0.54	0.40

Table 4.3: Recall

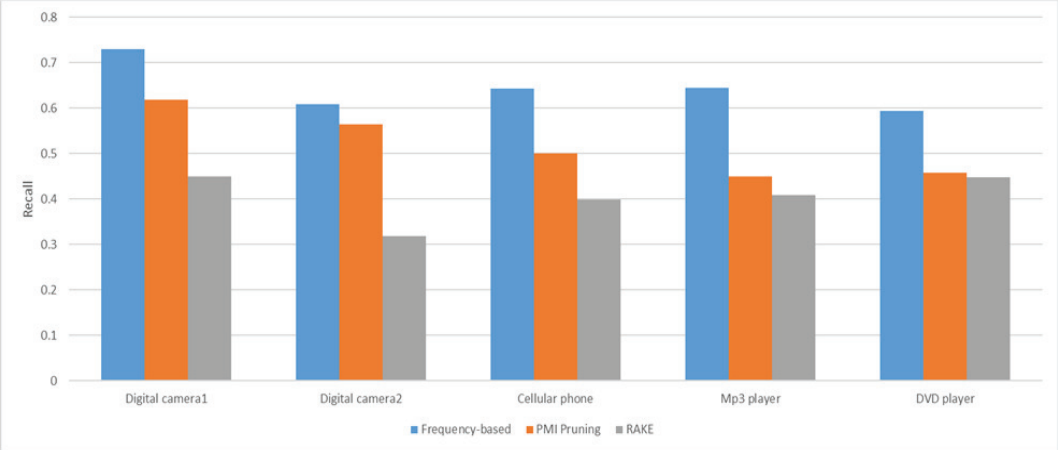

Figure 4.3: Recall

Dataset	NO. of Manual Features	Frequency-based	Web PMI-based	RAKE
Digital Camera1	89	0.75	0.70	0.53
Digital Camera2	69	0.72	0.70	0.35
Phone	98	0.70	0.63	0.45
MP3 Player	149	0.66	0.56	0.49
DVD Player	96	0.69	0.59	0.58
Average	100	0.70	0.64	0.48

Table 4.4: F-measure

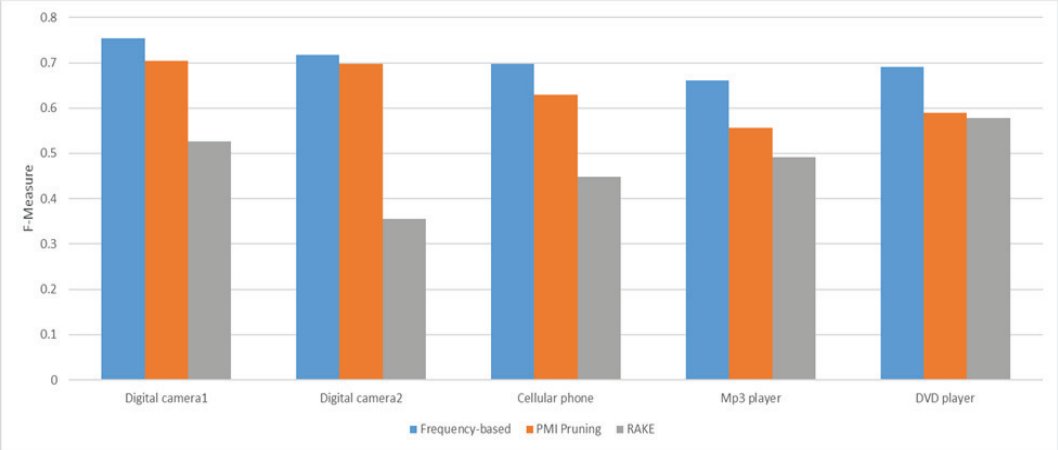

Figure 4.4: F-measure

4.2.2 Time Complexity Analysis

The time complexity of an algorithm is another significant performance indicator, which describes the amount of time to perform the task. The time complexity analysis can help us predict the time of the three methods.

For a given input of size n, the time complexity of the three feature word extraction methods can be expressed as follows:

- Frequency-based method: O(n^2)
- Web PMI-based method: O(n^2)
- RAKE method: O(n)

Table 4.5 gives out the execution time of each method. As shown in Figure 4.5, RAKE has a particular advantage in this analysis. This method does not need to parse the sentences using natural language analysis tools, which may cost a large amount of time. The implementation of the RAKE method is simple and quick. Thus, if the users want to get a quick scan of the reviews of one product, RAKE will definitely be a good choice.

On the other hand, although the time spent on querying the Web for hit counts is not included, the Web PMI-based method still takes the longest time. It is mainly because this method needs to read the hit counts from the local database for PMI score calculation. The Web PMI-based method not

only takes more time but also requires more space to store hit counts.

Dataset	NO. of Manual Features	Frequency-based	Web PMI-based	RAKE
Digital Camera1	89	261.57	284.04	5.53
Digital Camera2	69	118.41	234.08	3.08
Phone	98	214.11	248.55	4.16
MP3 Player	149	803.63	854.07	11.24
DVD Player	96	273.55	281.47	4.79
Average	100	334.26	380.44	5.76

Table 4.5: Runtime(s)

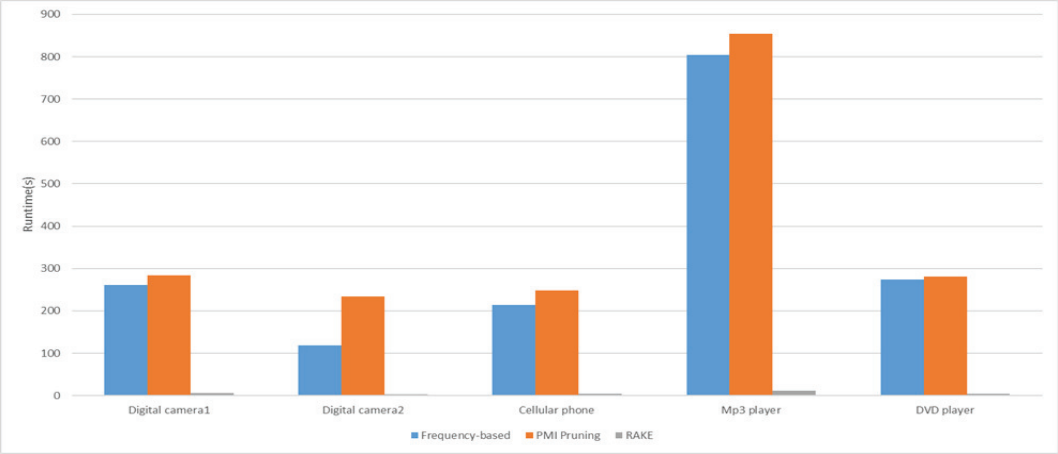

Figure 4.5: Runtime(s)

Note: Web PMI-based method's runtime were calculated without the time spent on querying the Web for hit counts.

4.2.3 Robustness Analysis

The robustness indicates the stability of an algorithm. We all know that the online text may contain some meaningless words, cyber terms, and trash information. A good feature word extraction algorithm can perform properly even if the input reviews contain some noises.

To test the robustness of each method, we randomly inject some automatically generated meaningless words and popular cyber terms into the datasets. The automatically generated meaningless words are random strings, e.g. "wtruyeadnx" and "yyuoa". The cyber terms are popular chat expressions, e.g. "lol", "gr8", and "btw". For each noisy word, we randomly pick up a number of sentences and insert the word into these sentences. Here the numbers of the noisy words and sentences are determined by the number of reviews in each dataset. We then conduct the same experiments. As shown in Table 4.6 and Table 4.7, it is clear that all the three methods have a good robustness to the noises.

For the frequency-based method and the Web PMI-based method, it is mainly because of the natural language processing, which works well in identifying the meaningless words and cyber terms by parsing and POS tagging. On the other hand, although the RAKE method does not have nature language analysis, this method has a simple noise filter, WordNet. This lexical database provides the POS tags of words, which helps identify the noun words as

candidates.

As a result, all of the three methods have a good performance in robustness analysis. They can be used to analyze disordered data sources with noises, such as Twitter.

Dataset	NO. of Manual Features	Frequency-based	Web PMI-based	RAKE
Digital Camera1	89	0.78	0.82	0.63
Digital Camera2	69	0.88	0.91	0.4
Phone	98	0.76	0.81	0.51
MP3 Player	149	0.68	0.70	0.62
DVD Player	96	0.83	0.83	0.81
Average	100	0.78	0.81	0.60

Table 4.6: Precision in Robustness Analysis

Dataset	NO. of Manual Features	Frequency-based	Web PMI-based	RAKE
Digital Camera1	89	0.73	0.62	0.45
Digital Camera2	69	0.61	0.57	0.32
Phone	98	0.64	0.59	0.40
MP3 Player	149	0.64	0.48	0.41
DVD Player	96	0.60	0.46	0.45
Average	100	0.64	0.54	0.40

Table 4.7: Recall in Robustness Analysis

Chapter 5

Rapid Feature Word Extraction Method

In this chapter, we propose a new approach based on the popular feature word extraction methods described before. Our goal is to improve the performance of these methods and to find a better new approach to extract feature words. To prove the advantages of the new approach, it is evaluated in the same way as shown in Chapter 4.

According to the experiments and analyses provided before, we have already known the pros and cons of the popular methods. The new approach, as a combination of these popular methods, is designed and developed based on the characteristics of these methods. In this way, we can make the most of

their strengths and get a better performance in feature word extraction.

5.1 Algorithm Design

As discussed before, most feature word extraction methods have three main steps to solve the problem. They are (1) candidate feature word recognition; (2) product-candidate correlation evaluation; (3) result pruning. As a feature word extraction method, the new approach has no exception.

In our study, many techniques are available in these three steps. For example, lexical analysis or lexical database can be used to recognize candidate feature words, such as Stanford CoreNLP and WordNet. Statistical methods can be used to evaluate the correlation between product and candidate feature words, such as word frequency and co-occurrence. There are also several pruning methods proposed to purify the results, such as compactness pruning, redundancy pruning, and Web PMI-based pruning. These techniques have their own characteristics. And our work is to find the most appropriate combination of the techniques used in the popular feature word extraction methods.

Thus, in order to achieve better performance, we need to analyze the pros and cons of the techniques first. The experiments in Chapter 4 provide a good chance for us to understand these techniques. Since all of the three pop-

ular methods perform well in robustness analysis, here we mainly concern about precision and recall analysis and time analysis. Through experiments on datasets, it is easy to find that the frequency-based method has a better performance in precision and recall analysis. Besides, since the RAKE method does not need any lexical analysis tool to parse sentences, it wins a landslide in time analysis.

Based on these observations, we decide to improve the speed of the frequency-based method by applying the parsing technique in the RAKE method, so that the new approach can ensure positive results in precision, recall, and runtime. The new approach is named as rapid feature word extraction method, or RFWE for short.

5.2 The RFWE Method

As a feature word extraction method, the RFWE method has three main steps without exception. Same as the frequency-based method, the RFWE method evaluates the weight of each candidate by calculating its frequency in reviews. Compactness pruning and redundancy pruning are also applied in the RFWE method to purify the results. On the other hand, RFWE identifies candidates by parsing sentences with punctuation marks and stop words.

Algorithm 5 RFWE Method

Require: *reviewslist*: a list of reviews
Ensure: *candidates*: a list of candidates after pruning

```
1: function RFWEMethod(reviewslist)
2:     candidates = new List(PhraseInfo)
3:     psupport = new Map(PhraseInfo, int)
4:     c = new PharseInfo()
5:     for word in reviewslist do
6:         if !WordNet.isNoun(word) or word.isStopword() then
7:             candidates.add(c)
8:             c = new PharseInfo()
9:         else
10:            c.add(word)
11:        end if
12:    end for
13:    for sentence in reviewslist do
14:        for c in candidates and c in sentence do
15:            if ∄ c': c' in candidates and c' in sentence and c in c' then
16:                psupport[c']++
17:            end if
18:        end for
19:    end for
20:    compact = new Map(PhraseInfo, int)
21:    for c in candidates do
22:        for sentence in reviewslist do
23:            if MaxWordsDistance(c, sentence) < 3 then
24:                compact[c]++
25:            end if
26:        end for
27:    end for
28:    for c in candidates do
29:        if (psupport[c] < 3 and ∃ c': c' in candidates and c in c') or
       compact[c] < 2 then
30:            candidates.remove(c)
31:        end if
32:    end for
```

(to be continued on next page)

Algorithm RFWE Method (continued from previous page)

```
33:     for c in candidates and psupport[c]> threshold do
34:         c.Weight = psupport[c]
35:     end for
36:     candidates.SortByWeight()
37:     return candidates
38: end function
```

The complete procedure of the RFWE method is shown in Algorithm 5. The method first extracts all nouns and noun phrases (lines 5-12 in Algorithm 5) and stores them into a list of candidates. Then it calculates the *p-support* (lines 13-19) and compact degree (lines 21-27) of each candidate. If the *p-support* of candidate c is smaller than 3 and c is a subset of another candidate *c'*, c is removed. Or if the compact degree of candidate c is smaller than 2, c is removed. After the pruning process, the remaining candidates are returned in an order of their weights (*p-support*).

5.3 Experiments

To validate the performance of RFWE method, we conduct experiments on the same datasets mentioned before. This section provides comments and analyses on the RFWE method, including comparison with the popular feature word extraction methods in precision, recall, F-measure, and runtime.

Table 5.1 shows us the scores of the RFWE method in precision, recall,

Dataset	NO. of Manual Features	Precision	Recall	F-measure	Time(s)
Digital Camera1	89	0.80	0.84	0.82	10.77
Digital Camera2	69	0.79	0.80	0.79	5.33
Phone	98	0.71	0.61	0.66	7.42
MP3 Player	149	0.72	0.83	0.77	23.40
DVD Player	96	0.72	0.57	0.64	8.55
Average	100	0.75	0.73	0.74	11.09

Table 5.1: RFWE Performance

	Frequency-based	Web PMI-based	RAKE	RFWE
Precision	0.78	0.81	0.60	0.75
Recall	0.64	0.54	0.40	0.73
F-measure	0.70	0.64	0.48	0.74
Time(s)	334.26	380.44	5.76	11.09

Table 5.2: Average Performance Comparison with the Popular Methods in Precision, Recall, F-measure, and Run Time

F-measure, and runtime. On the other hand, Table 5.2 gives performance comparison with the popular feature word extraction methods. Here we mainly compare the performance of the new method with the frequency-based method. As shown in Table 5.2, the RFWE method's recall is 9% higher than frequency-based method at a cost of 3% drop in precision. The RFWE method's F-measure score is the highest of all the methods, which means that

the RFWE method works better and has a high accuracy. Meanwhile, the RFWE method obviously takes less time than the frequency-based method. Put all these things together and compare with all the popular methods. It is clear that the new approach obtains a better performance in most cases and can be suitable for more applications.

Chapter 6

Conclusions

6.1 Summary

This thesis deals with text mining and describes three typical feature word extraction methods: the frequent-based method, the Web PMI-based method and the RAKE method. The goal is to give an objective evaluation on these popular methods and to propose a new approach based on the pros and cons of the popular methods. We conducted experiments and provided analyses on several aspects: precision, recall, F-measure, time complexity, and robustness. The analysis results are as follows.

If the users want a quick summary of the product features, the RAKE method is the best choice. Since this method extracts phrases with multiple words, it

includes more information than the other feature word extraction methods. RAKE is very easy and efficient in this case.

If the users want the summary more accurate, which means high precision, recall, and F-measure, the frequency-based method will be a good choice. This method has a better balance between precision and recall than the Web PMI-based method.

Last but not the least, all of the three methods perform well in robustness analysis. Thus, if the input data source contains some noises, the users can pick any method according to individual needs.

Based on these observations, we design the RFWE method, which combines the techniques used in the popular methods. The RFWE method uses the parsing technique in the RAKE method to identify candidates, which can increase the speed of the algorithm. Then the nouns and noun phrases with high frequencies in reviews are selected. Compactness pruning and redundancy pruning are applied in RFWE to prune the results. It is easy to find that the new approach performs well in precision and recall analysis, and time complexity analysis through experiments. Thus, the RFWE method is a great option for users to extract feature words in online reviews.

6.2 Future work

In our future work, we plan to further improve feature word extraction including the following topics:

- The thesis mainly focuses on finding explicit words that appear in customer reviews. We also need to find implicit feature words (Section 1.1). Some natural language processing techniques (e.g. semantic analysis) might meet our needs.

- Besides frequent features, there are some features that only a few customers are talking about. These infrequent features can also be important in some applications.

- Although the Web PMI-based method does not work as we expect, the Web is an immense and helpful corpus. The hit counts returned directly from Google search engine are not reliable enough. It is interesting to find that the hit count of one query in various browsers might be different. The hit counts also change over time. Thus, we need to find a better way to get the hit counts and improve the accuracy.

- We use meaningless words to test the robustness, which are easily filtered. More sets of noisy words need to test in the future, e.g. meaningful words.

Bibliography

[1] MCAULEY, J. AND LESKOVEC, J. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In Proceedings of the 7th ACM conference on Recommender systems, Anonymous ACM, 165-172.

[2] HU, M. AND LIU, B. 2004. Mining and summarizing customer reviews. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, Anonymous ACM, 168-177.

[3] PANG, B. AND LEE, L. 2008. Opinion mining and sentiment analysis. Foundations and trends in information retrieval 2, 1-135.

[4] QIU, G., LIU, B., BU, J. AND CHEN, C. 2011. Opinion word expansion and target extraction through double propagation. Computational linguistics 37, 9-27.

[5] POPESCU, A. AND ETZIONI, O. 2007. Extracting product features and opinions from reviews. In Natural language processing and text mining,

Anonymous Springer, 9-28.

[6] ROSE, S.J., COWLEY, W.E., CROW, V.L. AND CRAMER, N.O. 2012. Rapid automatic keyword extraction for information retrieval and analysis. U.S. Patent No. 8,131,735. 6 Mar. 2012.

[7] GROBELNIK, M., MLADENIC, D. AND MILIC-FRAYLING, N. 2000. Text mining as integration of several related research areas: report on KDD's workshop on text mining 2000. ACM SIGKDD Explorations Newsletter 2, 99-102.

[8] TAN, A. 1999. Text mining: The state of the art and the challenges. In Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases, Vol. 8. 1999.

[9] FELDMAN, R. AND SANGER, J. 2007. The text mining handbook: advanced approaches in analyzing unstructured data. Cambridge University Press.

[10] YI, J., NASUKAWA, T., BUNESCU, R. AND NIBLACK, W. 2003. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In Data Mining, 2003. ICDM 2003. Third IEEE International Conference on, Anonymous IEEE, 427-434.

[11] AGRAWAL, R. AND SRIKANT, R. 1994. Fast Algorithms for Mining Association Rules in Large Databases. In Proceedings of the 20th In-

ternational Conference on Very Large Data Bases, Anonymous Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 487-499.

[12] ZHUANG, L., JING, F. AND ZHU, X. 2006. Movie review mining and summarization. In Proceedings of the 15th ACM international conference on Information and knowledge management, Anonymous ACM, 43-50.

[13] DE MARNEFFE, M. AND MANNING, C.D. 2008. Stanford typed dependencies manual. Technical report, Stanford University, 2008.

[14] KOBAYASHI, N., INUI, K. AND MATSUMOTO, Y. 2007. Extracting Aspect-Evaluation and Aspect-Of Relations in Opinion Mining. In EMNLP-CoNLL, Anonymous Citeseer, 1065-1074.

[15] QIU, G., LIU, B., BU, J. AND CHEN, C. 2009. Expanding Domain Sentiment Lexicon through Double Propagation. In IJCAI, Anonymous, 1199-1204.

[16] ZHANG, L., LIU, B., LIM, S.H. AND O'BRIEN-STRAIN, E. 2010. Extracting and ranking product features in opinion documents. In Proceedings of the 23rd International Conference on Computational Linguistics: Posters, Anonymous Association for Computational Linguistics, 1462-1470.

[17] SCAFFIDI, C., BIERHOFF, K., CHANG, E., FELKER, M., NG, H. AND JIN, C. 2007. Red Opal: product-feature scoring from reviews. In

Proceedings of the 8th ACM conference on Electronic commerce, Anonymous ACM, 182-191.

[18] ETZIONI, O., CAFARELLA, M., DOWNEY, D., POPESCU, A., SHAKED, T., SODERLAND, S., WELD, D.S. AND YATES, A. 2005. Unsupervised named-entity extraction from the web: An experimental study. Artificial Intelligence 165, 91-134.

[19] BERRY, M.W. AND KOGAN, J. 2010. Text Mining. Applications and Theory.West Sussex, PO19 8SQ, UK: John Wiley & Sons.

[20] JONES, S. AND PAYNTER, G.W. 2002. Automatic extraction of document keyphrases for use in digital libraries: evaluation and applications. Journal of the American Society for Information Science and Technology 53, 653-677.

[21] WITTEN, I.H., PAYNTER, G.W., FRANK, E., GUTWIN, C. AND NEVILL-MANNING, C.G. 1999. KEA: Practical automatic keyphrase extraction. In Proceedings of the fourth ACM conference on Digital libraries, Anonymous ACM, 254-255.

[22] HULTH, A. 2004. Combining machine learning and natural language processing for automatic keyword extraction. Department of Computer and Systems Sciences [Institutionen for Data-och systemvetenskap], Univ.

[23] MANNING, C.D., SURDEANU, M., BAUER, J., FINKEL, J., BETHARD, S.J. AND MCCLOSKY, D. 2014. The Stanford CoreNLP

Natural Language Processing Toolkit. In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, June, Anonymous Association for Computational Linguistics, Baltimore, Maryland, 55-60.

[24] KAO, A. AND POTEET, S.R. 2007. Natural language processing and text mining. Springer Science & Business Media.

[25] HEARST, M.A. 1998. Automated discovery of WordNet relations. WordNet: an electronic lexical database 131-153.

[26] HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P. AND WITTEN, I.H. 2009. The WEKA data mining software: an update. ACM SIGKDD explorations newsletter 11, 10-18.

[27] WIKIPEDIA. 2014. Precision and recall — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Precision_and_recall&oldid=635850583.

Appendix A

Stopwords Table

a	currently	himself	next	so	via
able	d	his	nine	some	viz
about	dare	hither	ninety	somebody	vs
above	daren't	home	no	someday	w
abroad	day	hopefully	nobody	somehow	want
according	dear	how	non	someone	wants
accordingly	definitely	howbeit	none	something	was
across	described	however	nonetheless	sometime	wasn't
actually	despite	http	noone	sometimes	watch
adj	did	hundred	no-one	somewhat	watching
after	didnt	i	nor	somewhere	way
afterwards	didn't	i'd	normally	soon	we
again	different	ie	not	specified	we'd
against	directly	if	nothing	specify	week
ago	do	ignored	notwithstanding	specifying	welcome
ahead	does	i'll	novel	still	well
ain't	doesn't	i'm	now	sub	we'll
all	doing	immediate	nowhere	such	went
allow	done	in	o	sup	were
allows	don't	inasmuch	obviously	sure	we're
almost	down	inc	of	t	weren't
alone	downwards	inc.	off	take	we've
along	during	indeed	often	taken	what
alongside	e	indicate	oh	taking	whatever
already	each	indicated	ok	tell	what'll
also	edu	indicates	okay	tends	what's

although	eg	inner	old	th	what've
always	eight	inside	on	than	when
am	eighty	insofar	once	thank	whence
amid	either	instead	one	thanks	whenever
amidst	else	into	ones	thanx	where
amongst	elsewhere	inward	one's	that	whereafter
an	end	is	only	that'll	whereas
and	ending	isnt	onto	thats	whereby
another	enough	isn't	opposite	that's	wherein
any	entirely	it	or	that've	where's
anybody	especially	it'd	other	the	whereupon
anyhow	et	it'll	others	their	wherever
anyone	etc	its	otherwise	theirs	whether
anything	etc.	it's	ought	them	which
anyway	even	itself	oughtn't	themselves	whichever
anyways	ever	ive	our	then	while
anywhere	evermore	i've	ours	thence	whilst
apart	every	j	ourselves	there	whither
appear	everybody	just	out	thereafter	who
appreciate	everyone	k	outside	thereby	who'd
appropriate	everything	keep	over	there'd	whoever
are	everywhere	keeps	overall	therefore	whole
aren't	ex	kept	own	therein	who'll
around	exactly	know	p	there'll	whom
as	example	known	particular	there're	whomever
a's	except	knows	particularly	theres	who's
aside	f	l	past	there's	whose
ask	fact	last	people	thereupon	why
asking	fairly	lately	per	there've	wife
associated	far	later	perfect	these	will
at	farther	latter	perhaps	they	willing
available	few	latterly	placed	they'd	wish
away	fewer	least	playing	they'll	with
awfully	fifth	less	please	they're	within
b	first	lest	plus	they've	without
back	five	let	possible	thing	woman
backward	followed	let's	presumably	things	women
backwards	following	like	probably	think	wonder
be	follows	liked	provided	third	won't
became	for	likely	provides	thirty	world
because	forever	likewise	q	this	would
become	former	little	que	thorough	wouldn't
becomes	formerly	long	quite	thoroughly	x
becoming	forth	lot	qv	those	y
been	forward	love	r	though	year
before	found	-lrb-	rather	three	yes
beforehand	from	ltd	rd	through	yet
begin	further	m	re	throughout	you
behind	furthermore	mainly	really	thru	youd

being
believe
below
beside
besides
best
better
between
beyond
big
both
brief
but
by
c
came
can
cannot
cant
can't
caption
cause
causes
certain
certainly
changes
clean
clearly
c'mon
co
co.
com
come
comes
concerning
consequently
consider
considering
contain
containing
contains
corresponding
could
couldn't
course
c's
g
gaming
get
gets
getting
given
gives
go
goes
going
gone
good
got
gotten
great
greetings
h
had
hadn't
half
happens
hardly
has
hasn't
have
haven't
having
he
he'd
he'll
hello
help
hence
her
here
hereafter
hereby
herein
here's
hereupon
hers
herself
he's
hi
high
him
make
makes
man
many
may
maybe
mayn't
me
mean
meantime
meanwhile
men
merely
might
mightn't
mine
minus
miss
month
more
moreover
most
mostly
mr
mrs
much
must
mustn't
my
myself
n
name
namely
nd
near
nearly
necessary
need
needn't
needs
neither
never
neverf
neverless
nevertheless
new
reasonably
recent
recently
regarding
regardless
regards
relatively
respectively
right
round
-rrb-
s
's
said
same
saw
say
saying
says
second
secondly
see
seeing
seem
seemed
seeming
seems
seen
self
selves
sensible
sent
serious
seriously
seven
several
shall
shan't
she
she'd
she'll
she's
should
shouldn't
since
six
thus
till
time
tis
to
today
together
tonight
too
took
top
toward
towards
tried
tries
truly
try
trying
t's
twas
twice
two
u
un
under
underneath
undoing
unfortunately
unless
unlike
unlikely
until
unto
up
upon
upwards
us
use
used
uses
using
usually
v
various
versus
very
you'd
you'll
your
you're
yours
yourself
yourselves
youve
you've
z
zero
'
"
-
!
"
#
$
%
&
(
)
*
,
.
/
:
;
?
@
[
]
^
()
`
{
|
}
~
()
+
=

Vita

Candidate's full name: Ting Zhang

University attended:
September 2013 - May 2016
Faulty of Computer Science
University of New Brunswick
Fredericton, New Brunswick, Canada

September 2008 - May 2012
Bachelor of Software Engineering
Software Engineering
Southeast University
Nanjing, Jiangsu, China

Publications:
Conference Presentations: