A framework for finding and summarizing product defects, and ranking helpful threads from online customer forums through machine learning

Jian Jiao

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

Doctor of Philosophy
in
Computer Science and Applications

Weiguo Fan, Chair
Alan Abrahams
Edward A. Fox
Naren Ramakrishnan
Gang Wang
Liqing Zhang

April 1, 2013
Blacksburg, Virginia

Keywords: Product Defect Detection, Product Feature Extraction, Summarization, Clustering, Learning to Rank, Thread Ranking

A framework for finding and summarizing product defects, and ranking helpful threads from online customer forums through machine learning

Jian Jiao

(ABSTRACT)

The Internet has revolutionized the way users share and acquire knowledge. As important and popular Web-based applications, online discussion forums provide interactive platforms for users to exchange information and report problems. With the rapid growth of social networks and an ever increasing number of Internet users, online forums have accumulated a huge amount of valuable user-generated data and have accordingly become a major information source for business intelligence. This study focuses specifically on product defects, which are one of the central concerns of manufacturing companies and service providers, and proposes a machine learning method to automatically detect product defects in the context of online forums. To complement the detection of product defects, we also present a product feature extraction method to summarize defect threads and a thread ranking method to search for troubleshooting solutions. To this end, we collected different data sets to test these methods experimentally. The results of the tests show that our methods are very promising. In fact, in most cases, they outperformed current state-of-the-art methods.

Acknowledgments

I would like to acknowledge and express my heartfelt gratitude to the following people who have made the completion of this dissertation possible.

- First and foremost, my supervisor, Dr. Weiguo Fan, for his valuable guidance and advice throughout my work toward a Ph.D. He inspired me to focus on online communities and devote efforts to important topics that can deliver true value in both research and practice. Besides, he was always helpful and offered invaluable assistance in my life.
- Drs. Edward Fox and Naren Ramakrishnan, for teaching me information retrieval and data mining and getting me involved in several important research projects, which greatly extended my knowledge and skills.
- Drs. Alan Abraham, Gang Wang and Liqing Zhang, for their guidance and assistance in research and paper publication.
- Special thanks to my family and friends for their support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As one of the most important applications in the Web 2.0 era, online forums have grown dramatically since they were launched. Today, as people are well served by these newly emerged applications, online forums are widely accepted by users (*Lynch*, 2009) and are becoming ever more influential in social and commercial contexts (*Dellarocas*, 2006; *Stodder*, 2010a,b). Users post observations and exchange knowledge about products and services on the Internet. Immediately, these data are available to millions of people all over the world, and user input quickly propagates through social media. User-generated information from customer forums has become a vital factor determining the fate of a company, so proper extraction and smart use of this information is now one of the most effective ways for a company to gain competitive advantages and adapt in times of change.

To understand and analyze user-generated information, a group of tools and methodologies have been developed and extensively employed in practice. These include data mining (*Berry and Linoff*, 1997), business intelligence (BI) (*Liautaud*, 2000), customer relationship management (CRM) (*Slater and Narver*, 2000), and data warehousing (*Inmon*, 1992). Traditionally, the raw data used by these systems have been user transactions rather than user interactions. Undoubtedly, transaction data will continue demonstrating its value in the future. However, with the rapid development of online customer forums, the data collected therein have started to reveal their great potential to different parties. Online customer forums are places where users from all over the world get together, discuss product-related topics, seek help for troubleshooting problems and share experiences related to certain products. Typically, a customer forum is created exclusively for a certain product or a certain category of products, officially or unofficially. Compared to customer transactions, online forums have the following advantages (*Stodder*, 2010a,b): 1) Instead of individual behaviors, social networks make it possible to analyze the behaviors of small communities and the interactions among the community members. 2) The unstructured texts and semi-structured data in online customer forums provide more possibilities and more dimensions to investigate. 3) The voice of users might be biased in other media like blogs and online reviews, which are

authored by single customers; in online forums, however, users can freely communicate with others and therefore more reliable knowledge comes out in the process. 4) Online forums as a data set have great potential. As the Internet continues to grow, online forums will generate more traffic and attract more users, thereby becoming more important to business and service provides.

A piece of news in 2010 trained a spotlight on product defects for automobile customers far and wide. Toyota Motor issued the second massive recall in two months due to problems with a sticking accelerator pedal (*Bunkley*, 2010; *Healey and Carty*, 2010), which involved 2.3 million Toyota-brand cars and trucks in the U.S. dating back to 2005 models. Together with an earlier recall by Toyota to fix a floor mat entrapment problem (*Tabuchi and Maynard*, 2009), which included 3.8 million vehicles, the loss brought about by these two product defects was estimated to be upwards of $2 billion (*Isidore*, 2010). With these two recalls, the nightmare for Toyota did not come to an end, however. According to another piece of news (*Lea*, 2010), Toyota had to temporarily close its plants in the U.S. and was expecting a massive drop in sales, a turn of events that presented a huge opportunity to its competitors to fill the market gap and lure away Toyota's customers. Furthermore, the U.S. government also got involved, as the National Highway Traffic Safety Administration (NHTSA) faced tough questions from customers (*Valdes-Dapena*, 2010):"NHTSA failed the taxpayers, Toyota failed its customers." The NHTSA came under criticism for not acting soon enough to protect auto consumers. A long time before Toyota issued its recalls, the NHTSA could have accumulated sufficient evidence of fatal crashes and plentiful complaints to step in and investigate the problem; the agency, however, was not that vigilant or active in its role as a consumer watchdog.

Similarly, product defects also incurred substantial losses in other industries. In 2006, Dell recalled 4.1 million laptop batteries (*Vance*, 2006), which was estimated to cost Dell and its collaborative partner Sony $400 million (*NYTimes*, 2006). According to statistical recall data from WeMakeItSafer, there has been an increasing number of product recalls in the 2000s (Figure 1.1) , with a peak in product units recalled in 2004 (Figure 1.2).

Table 1.1: Number of incidents reported at time of recall by year

Year	Incidents w/o harm	Property damage	Fires	Injuries w/o deaths	Deaths
2004	1,815	200	240	535	2
2005	5,550	665	155	810	0
2006	4,580	185	165	490	7
2007	12,460	305	395	965	6
2008	3,670	170	360	565	7

Table 1.1 lists the incidents and injuries caused by these recalled products at the time that the recalls were made. These numbers do not represent the total incidents attributable to faulty products. Instead, more incidents occurred after the recalls were made. Still, had better product defect alert systems been in place, many of these incidents might have been

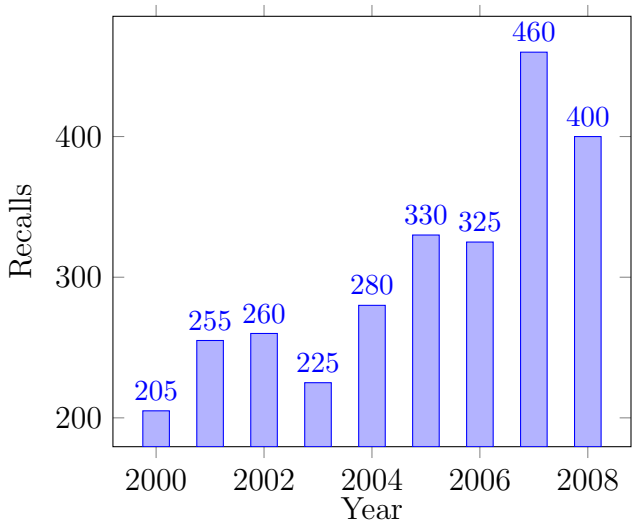

Figure 1.1: Customer product recalls by year

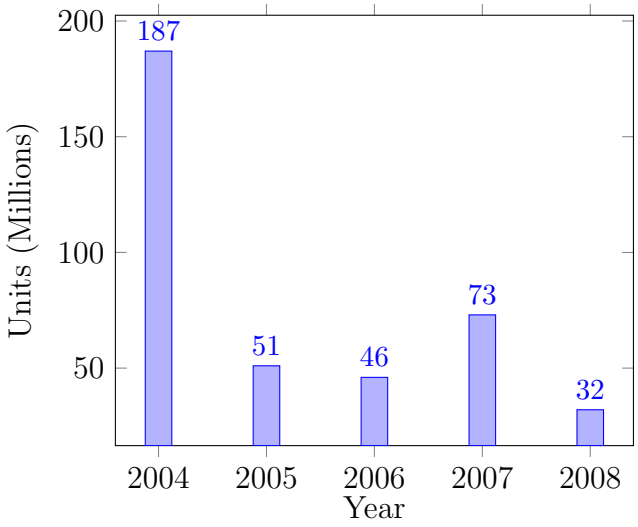

Figure 1.2: Number of recalled units by year

prevented. What these numbers tell us is that product defects are far from trivial matters for both consumers and companies. On the contrary, they are crucial factors to companies' success and to consumers' satisfaction and safety.

Different from other business intelligence systems, which were intended to analyze customer intelligence and behaviors using data from online forums (*Berry and Linoff*, 1997; *Liautaud*, 2000; *Slater and Narver*, 2000), the focus of my research is on product defects. Product defects are a major concern for customers and constitute some of the most popular topics in user forums. From the perspective of companies that manufacture and sell consumer products, identifying product defects is essential to attracting and retaining customers, improving products, remaining competitive, and preventing customer drift to competitors' products. Traditionally, reports of product defects have been collected and processed by persons other than consumers themselves, for example, analysts in a company's customer service department and a government consumer safety agency. Such defect reporting systems are costly, labor intensive, time consuming and sometimes slow to act. For industries like auto manufacture, late reporting of a product defect may cause a severe accident or a big profit loss. As a result, timely, automated product defect identification and processing is a technological capability that is in high demand.

In this dissertation, we discuss product defects in the context of online forums and develop an effective and comprehensive framework to address three related problems: product defect detection, product feature extraction, and authoritative thread ranking. These problems correspond to three of the most challenging tasks in the process of identifying, summarizing, and correcting product defects.

The first research task of this study, the problem of product defect detection, is formulated as a binary classification problem. To precisely characterize defect discussion threads, hundreds of features of different types are extracted and a heuristic feature selection method is adopted to find the optimal subset of features. Experimental results show that a support vector machine (SVM) classifier trained on selected features can effectively identify defect threads in online forums. And within this same framework, the criticality of defects also can be investigated, with safety and performance defects separated from each other.

The second task is product feature extraction. Using a machine learning approach, the product features in each thread of a customer forum are effectively extracted and then properly grouped into product feature clusters. The results of the first sub-task are compared to some other unsupervised and supervised learning methods, and our proposed method achieves better performance in both precision and recall. In the second sub-task, the proposed product feature clustering method with combined distance function integrates information from different angles and performs the best. Noteworthy, too, is that the same product feature extraction method also can be adopted in a different field for a different purpose. For example, it can be combined with sentiment analysis to produce a list of product features with customer opinions on products.

The last research problem is helpful thread ranking, which helps to ascertain the best solu-

tion when troubleshooting a problem, if indeed the matter at issue is solvable. Two types of features are collected as indicators of argument quality and source credibility. Genetic algorithm (GA) and genetic programming (GP) methods are adopted to optimize and search for the optimal ranking formula based on new fitness functions. To evaluate the performance of proposed algorithms and meanwhile test the generality, two different forums are collected. Results from experimental tests show that the proposed method of thread ranking can effectively find threads that are not only highly related but also authoritative and helpful.

This study could bring benefits to the following parties:

- Manufacturing companies and service providers

  Based on the framework presented in this dissertation, a business intelligence system could be developed that benefits manufacturers and service providers by providing product design feedback (leading to innovation and improvement) and improved customer service and public relations. This in turn could increase sales and revenue, and perhaps reduce costs resulting from fines and penalties imposed for a lack of due diligence.

- Customers

  A summary of emergent product defects could guide potential customers in their selection of products from the category of products available for purchase. Moreover, authoritative threads or solutions on product defects can provide positive suggestions to help customers resolve defects in the products they already own. Last but not least, by generating awareness of product defects, customers might make companies more responsive to their needs.

- Industry

  A defect discovered in one product may help prevent similar defects in other products and potentially improve the product quality across an entire industry.

- Government

  As the ones who put products to use, customers are often the first to discover product defects after products are released. By analyzing the product defects discussed by customers in Web forums, a warning system can be devised to help the government monitor product defects, improve job efficiency and build up taxpayer confidence in government agencies.

- Academia

  The novel algorithms and frameworks proposed in this dissertation may broaden the research horizon in data mining and information searching and bring improvements to existing techniques: Machine learning algorithms are used to solve the problem of product defect identification. Secondly, an effective supervised learning approach is

presented to summarize product related forum threads. Lastly, learning to rank is applied to address the thread ranking problem in online forums.

To conclude, although such a system is very critical and beneficial to different parties, it did not receive enough attention. Our work will fill this gap.

The rest of this dissertation is divided into three parts each of which focuses on one of the research problems: Chapter 2 concentrates on product defect detection and defect criticality including a literature review, an explanation of methodology, and a description of experiments. Chapter 3 discusses product feature extraction and feature clustering methods in detail. Chapter 4 covers the proposed solution for thread ranking problem–authoritative thread ranking.

Chapter 2

Product Defect Detection

Online communities have become a very important source of business intelligence. They can provide information about product defects, which are a major concern of manufacturing companies and service providers. Is there a way to find defect-related information from an online forum? And, if so, how does one automatically detect this information? In this chapter, we look into product defects identified in online communities and propose a supervised classification method to automatically discover defects and systematically determine their level of criticality. According to the experimental results, product defects can be effectively detected and classified according to proper criticality levels. Beyond that, the proposed method is generalizable: once trained in one product forum, the same model can be easily adapted to other product forums in the same industry.

2.1 Introduction

Nowadays, communicating with others in online communities has become commonplace in many people's daily lives. Online communities attract people with common interests from all over the world and provide a platform for them to post ideas and exchange opinions. In online product forums, the common interest is brands and products. With the growth of online communities, large volumes of knowledge are accumulated, and no business or organization can afford to ignore their great informational value. Online communities have therefore become a very important source of business intelligence and have caught the attention of researchers.

For any business that designs and manufactures goods, product quality is of the utmost importance for commercial success. Detecting defects can be a great start, no matter whether the goal is to ensure or to improve product quality. Discovery of product defects can help a company improve its products and remain competitive in the marketplace. Traditionally, product defects are detected through a series of carefully devised inspections and tests. How-

ever, these tests cannot discover all the potential problems of products, especially problems that only develop with use over time. It is usually very expensive and requires a lot of effort to collect such information. An alternative is to leverage customer experience as a resource of business intelligence, as customers know the products well and have put them to use extensively.

Online communities present just such a source of customer feedback. In online communities, many customers get together and post their experiences of using products or services. As they communicate, one topic sure to arise is that of product faults and defects. Participants in online communities have different relationships to and uses of products, so their online postings offer a variety of representative experiences. Furthermore, online forums constitute large data sets, which make them rich sources for analysis. All these characteristics together make online communities a good data source for defect detection.

To demonstrate what a defect thread looks like, the following is an example defect thread from the Honda-tech forum.

> ***title**: 2004 accord metal grinding sound*
> ***content**: I just got a 2004 accord coupe v6 6 speed 2 weekends ago. There is some sort of metal grinding sound coming from the front passenger side of the car where the belts are. I do not know what the sound is coming from or what is causing it. The noise happens just above 2000 rpms and i am accelerating. The noise happens in every gear but it does not happen when i am just reving the car. When i coast at 2000rpms it doesn't happen but when i put my foot on the gas it happens. I searched around and read that it could be something with the brakes or the exhaust heat shield is loose. Any suggestions would be nice, so i can trouble shoot before i take it to a dealer and spend money to find out what is wrong. Thanks.*

The above thread talks about a grinding sound that is not due to a misused or a worn-out auto part, but instead is a transmission defect of the 2004 Honda Accord (see recall (*MotorTrend*, 2012)). The thread is evidence of defect identification in a forum, but there are still lots of other discussions. Actually, the topics of threads change a lot. Some people may chat about overall experiences or feelings about products or brands, whereas others may ask about product usage or pricing. Specific to the interest of this work, topics can be generally grouped into two categories: discussions regarding product defects in design or manufacturing, and non defect-related discussions, which are mostly concerned with user-specific problems, such as misuse and wear and tear. Although all user-generated content has some value, identification of true product defects is critically important to companies, especially when the information comes directly from their customers. How to automatically find defects from among millions of threads is an imperative but challenging task. To the best of our knowledge, little study of this matter has been reported in the research literature to date. Considering the great value of defect information for businesses, in this chapter we

concentrate on discovering product defects from online communities to help guide companies in their efforts to improve the product quality.

Beyond that, we are also interested in the criticality level of product defects. Two levels of criticality are defined in the context of this study: safety critical and performance critical. The categories of safety and performance may have different meanings depending on the product in question. For example, if a problem with an automobile causes a fire or an injury, then it must be very critical and should be categorized as a safety critical defect. On the other hand, a defect such as a car's radio not working properly should be categorized as a performance detect. In the case of the Apple forum, a defect resulting in the crash of a computer system should be classified as a safety critical defect, while a defect like poor sound quality in a mobile device is a performance defect. In our research, we seek to find a generalizable and effective algorithm that works for this particular research problem and that can be applied in different forums or even in different industries.

According to Merriam-Webster's dictionary, the definition of a defect is

> *(1) an imperfection that impairs worth or utility (2) a lack of something necessary for completeness, adequacy, or perfection.*

From the above definition, we can see that a product defect has two properties: (1) it is critical and has an impact on utility; (2) a product defect may hurt a customer's satisfaction. With this basic definition in mind, we give a formal definition of a product defect as follows:

Definition 1. *A* ***product defect*** *is a critical problem or malfunction in a certain product or service that concerns the customers and stimulates a discussion in the customer forum. A product defect has two criticality levels:* ***safety critical defects*** *and* ***performance critical defects****. The former denotes a defect that results in a system crash or a serious accident and generates a big impact, whereas the latter denotes other defects related to system performance that are not as critical as safety critical defects. The two criticality levels and non-defect issues are together regarded as three thread categories denoted as* $c_i \in \{1, 2, 3\}$*, where* $c_i = 1$ *means non-defect issue thread,* $c_i = 2$ *means performance critical thread, and* $c_i = 3$ *means safety critical thread.*

To summarize, the product defect detection problem can be formulated as the following question:

In an online customer forum, what are the defect-related threads and how critical are they?

From the above formulation, we can see that it is actually a multi-class classification problem, where each thread is associated with no more than one class. The proposed approach will solve it in two steps: first, by identifying the defect threads and, then, by predicting the criticality levels of the defects. Details about the methodology and experiments are discussed in the remainder of this chapter.

2.2 Literature review

To the best of our knowledge, this study is the first attempt at using a text classification method to solve the defect detection problem; it is based on data from online forums, which makes it different from traditional classification. In this section, we review relevant past research from three perspectives: classification tasks using online data, classification techniques, and feature sets.

2.2.1 Classification tasks using online data

Online content includes emails, blogs, forum posts, tweets, images, audio and video files (*Morville and Rosenfeld*, 2006). The development of Web 2.0 technologies and the growth of Web content have attracted the attention of many researchers, who have sought ways to search and sort these rich data sources effectively. The research problems that need to be resolved using text classification techniques include authorship identification, sentiment analysis, gender classification and post/thread quality assessment.

Online content is widely used in different situations and broadcast across the Internet, but sometimes the content is improperly solicited or intentionally unauthorized for certain purposes. De Vel and his colleagues proposed to use a text classification method to identify the original authors of emails (*De Vel*, 2000; *De Vel et al.*, 2001). Two types of features, style features and structure features, are employed in their work, and both of them proved to be effective. Their experimental results were encouraging but might not be significant enough since the size of the data set was relatively small (the data set only contained works from three authors). Zheng et al. developed a framework with four types of features and their approach effectively identified authors with satisfactory accuracy on a mixed data set in both English and Chinese (*Zheng et al.*, 2006). A recent study was conducted by Abbasi and Chen (*Abbasi and Chen*, 2008), in which they applied their newly developed technique, Writeprints, with a richer set of features. Their experimental results using a data set of 100 authors were very promising.

A broad branch of research using online data is sentiment analysis. Inspired by the rating system of some online review websites, people are seeking to understand and predict the opinions and sentiments of others simply based on the texts they generate. Opinion Finder (*Wiebe et al.*, 2005; *Wilson et al.*, 2005) is one such system that automatically identifies the subjective sentences from a text and determines the author's opinion orientation as positive, neutral, or negative. A naive Bayes classifier was first trained to discriminate a sentence as subjective or objective using a variety of lexical and contextual features. In the end, the user's opinion orientation was identified using BoosTexter, a general purpose boosting classifier.

Gender classification is a machine learning task that aims to identify whether a piece of

writing is produced by a female or a male mainly through textual analysis. The difference between authorship identification and gender classification is that the former needs to distinguish the writing style of each individual author whereas the latter needs to identify the general style of a group of authors. Some of the early research in this area was conducted mainly based on traditional documents like fiction and nonfiction (*Koppel et al.*, 2002). Later researchers changed the focus to emails (*Corney et al.*, 2002) and blogs (*Nowson and Oberlander*, 2006), and most recently, online communities (*Zhang et al.*, 2011). Interesting patterns of gender difference were found in these works, i.e., Internet usage, interested topics, etc.

Data in online communities accumulate very fast, and due to a lack of editorial control, much user-generated content can be low in quality. Therefore, assessing the quality of online content is an important issue in many online communities, and many applications have been developed to address it, such as (*Weimer et al.*, 2007; *Baldwin et al.*, 2007).

One study conducted in the business domain, (*Coussement and Vandenpoel*, 2008), focused on identifying customer complaints from emails and presents a successful example of using online data and text mining methods to solve a practical business intelligence problem.

2.2.2 Classification techniques

Supervised classification is a classical machine learning task and is concerned with assigning a class label to a certain object on the basis of a set of objects with known labels. Simply put, if the size of the label set is two, it is called binary classification, whereas cases of a label size larger than two are called multi-class classification. The focus of this dissertation is binary classification. Various algorithms have been proposed to solve the binary classification problem including neural networks, decision trees, k-nearest neighbor, naive Bayes, and support vector machine (SVM).

Naive Bayes (*Rish*, 2001) is a simple probabilistic classification model based on Bayes' theorem. Naive Bayes classifiers work well in many complex situations. (*Baldwin et al.*, 2007; *Pang et al.*, 2002)

Logistic regression is a type of regression analysis used to predict the probability of a categorical outcome based on a set of dependent variables (*Hosmer and Lemeshow*, 2000). To leverage the probability character, the researchers developed a multi-class classification algorithm named multinomial logistic regression to solve the author identification problem (*Madigan et al.*, 2005b,a).

A decision tree is a tree-like graph, which leads users through the nodes by asking them predefined questions and letting them make appropriate decisions toward the goal, which by nature can support strategy identification and decision making. In data mining, a decision tree is a prediction algorithm that can learn based on extracted properties from training data sets and predict the target variable on unknown data sets(*Breiman et al.*, 1984). (*Abbasi*

and Chen, 2005; *Zheng et al.*, 2006) employed a decision tree algorithm.

Support vector machine (*Gunn*, 1998; *Hearst et al.*, 1998) is a very popular supervised learning model and has been adapted to solve many binary classification problems in online scenarios (*De Vel*, 2000; *Zheng et al.*, 2006; *Pang et al.*, 2002; *Weimer et al.*, 2007). The basic concept of SVM is to minimize the structure risk and generalization error. Compared to other models, SVM offers some important advantages: it is fairly robust to the over-fitting problem and can be easily adapted to considerable dimensions (*Sebastiani*, 2002). Therefore, in most cases, SVM out-performed other learning models (*Yang and Liu*, 1999).

Neural networks are networks composed of interconnected artificial neurons and perform like real biological systems. They are designed to work in a manner analogous to the human brain. Neural networks have been successfully used in different areas for solving artificial intelligence problems (*Bishop*, 1995; *Lippmann*, 1987; *Dagan et al.*, 1997). They provide an analytical alternative to conventional algorithms in that they can be applied to solve some complicated problems in which the assumption of normality, linearity, or variable independence is not very clear. In (*Zheng et al.*, 2006), neural networks were adopted to solve the author identification problem and produced results comparable to SVM. In the extreme case, logistic regression can be viewed as a two-layer neural network.

2.2.3 Features

The features that have been used in previous research can be grouped into four major categories (*Abbasi and Chen*, 2008): content-specific features, lexical features, syntactic features, and structure features. In most cases, a subset of features is carefully selected from one or more of these categories depending on the concrete problem needing to be solved.

Content-specific features are the most widely used features. They refer to important keywords or phrases in the domain (*Abbasi and Chen*, 2008). Unigram, bigram, and n-gram are the common forms of content-specific features. In previous studies, the content-specific features were either a small amount of manually selected domain keywords (*Abbasi and Chen*, 2008) or a large amount of words with relatively high document frequency (e.g., terms that appeared more than ten times in the document set(*Zhang et al.*, 2011)). Both methods have limitations: the former requires lots of domain knowledge and manual work, while the latter may include many stop words as features, which would not contribute much to classification. In our work, we present a systematic way of selecting content features.

To express the same idea, people have different preferences about which words to use. Lexical features are a category of features related to wording, and these features are expected to reflect an author's writing style. As a result, lexical features were extensively used in author identification (*De Vel*, 2000; *De Vel et al.*, 2001; *Abbasi and Chen*, 2008) and gender classification (*Corney et al.*, 2002; *Zhang et al.*, 2011).

Syntactic features relate to how different terms are assembled to form sentences; they include

punctuation, functional words and part-of-speech (POS) features. Considering the fact that syntactic features can capture authors' writing styles to a certain extent at the sentence level, most of the time they were used together with lexical features in author identification and gender classification.

Different from normal unstructured texts, online data are semi-structured and have more metadata, which can potentially boost the performance of machine learning algorithms. Structure features in an email corpus include the existence of a signature, number of attachments, number of quotations, etc. (*Corney et al.*, 2002). In online forums, user discussions are organized in messages/threads, and the following features are proposed based on thread structure (*Zhang et al.*, 2011)): number of sentences per message, number of paragraphs per message, number of sentences per paragraph in a message, etc.

There are also some other features previous researchers have used that do not fall into any of these categories. For instance, to assess post quality in online forums, a special feature named similarity was employed in (*Weimer et al.*, 2007), which was a measurement of the relatedness of a post to the topic of the sub-forum to which it belonged. Higher similarity was considered to indicate higher post quality. These types of features might not be generalizable, but sometimes, since they are tightly connected to the domain or problem, they can be powerful in practice.

Sometimes the feature space can be very large, with hundreds of content-free features and tens of thousands of content-specific features. Simply throwing all of them into the classifier is definitely not a good idea. On the other hand, it is not realistic to manually select good features from them. Feature selection is a key topic in machine learning (*Dash and Liu*, 1997; *Guyon and Elisseeff*, 2003) and has been widely used in many fields, such as data mining (*Liu and Yu*, 2005) and information retrieval (*Fan et al.*, 2005). The goal of feature selection is to remove irrelevant and redundant features; as a result, training and utilization are sped up and system performance is improved.

Existing feature selection algorithms fall into two broad categories: feature ranking and subset selection. The former is typically used to handle large numbers of term features in text classification (*Yang and Pedersen*, 1997) and information retrieval (*Fan et al.*, 2005), where the features are presumably independent of each other. Some key feature ranking methods are listed in Table 2.1.

According to (*Liu and Yu*, 2005; *Somol et al.*, 2010), existing subset selection methods can be roughly grouped into four categories with respect to the actual criteria used:

- The *filter method* (*Yu and Liu*, 2003) directly relies on the training data set itself to derive a subset of features with high correlation to the labels or a subset of features that, together, has the best evaluation properties, such as consistency and dependency.
- The *wrapper method* (*Kohavi and John*, 1997) is used to find a subset of features best suited to a predetermined machine learning method. Therefore, different algorithms

Table 2.1: Feature selection methods used in previous literature

Feature selection methods	Literature
Information gain (IG)	(*Dash and Liu*, 1997; *Fan et al.*, 2005; *Yang and Pedersen*, 1997)
Mutual information (MI)	(*Yang and Pedersen*, 1997)
Chi-square (CS)	(*Fan et al.*, 2005; *Yang and Pedersen*, 1997; *Ng et al.*, 1997)
Document frequency (DF)	(*Yang and Pedersen*, 1997)
Document and relevance correlation (DRC)	(*Fan et al.*, 2005)
Robertson's selection value (RSV)	(*Fan et al.*, 2005)
F-score (FS)	(*Chen and Lin*, 2006)

may produce different sets of features.

- The *embedded method* (*Guyon and Elisseeff*, 2003) integrates the feature selection process into the algorithm training. The best feature set is produced as a by-product of the learning method results. This can only be used with certain machine learning algorithms.
- The *hybrid method* (*Das*, 2001) combines the advantages of the filter method and the wrapper method, which can generate high-quality features with low time complexity.

In our work, a hybrid method was adopted to select a small set of features with high correlation to labels based on information gain scores and then search for an optimal subset of features that can potentially achieve an optimal performance using the wrapper method.

Tables 2.2 and 2.3 summarize the related previous work.

2.3 Research method

The system architecture of product defect detection is shown in Figure 2.1. The system consists of three main components: data collection, feature generation, and classification and evaluation. Online communities provide a large repository of user-generated content that includes a reasonable amount of product defect information. The responsibility of the *data collection* component is to crawl user posts in the form of HTML pages from discussion forums, parse out the post content and other necessary metadata, and import into the forum database.

In the *feature generation* component, different types of features are extracted through a series of processing steps including sentiment analysis, POS tagging, and n-gram indexing. In general, there are two broad types of features extracted: content-specific features and

Table 2.2: Summary of related work using online data: techniques and features

Literature	Techs	Features				
		Lexical	Syntactic	Structure	Content	Other
(*De Vel*, 2000)	SVM	√	√			
(*De Vel et al.*, 2001)	SVM	√	√	√		
(*Zheng et al.*, 2006)	SVM	√	√	√	√	
(*Abbasi and Chen*, 2008)	SVM	√		√	√	
(*Pang et al.*, 2002)	SVM	√	√	√		
(*Wilson et al.*, 2005)	NB	√	√	√		
(*Corney et al.*, 2002)	SVM	√	√	√		
(*Zhang et al.*, 2011)	SVM	√	√	√	√	
(*Weimer et al.*, 2007)	SVM	√	√	√		√
(*Baldwin et al.*, 2007)	MC	√	√	√	√	√
(*Coussement and Vandenpoel*, 2008)	Boost	√	√		√	
(*Abrahams et al.*, 2013, forthcoming)	SVM				√	

Table 2.3: Summary of related work using online data: tasks and data sets, including six research tasks: author identification (AI), opinion mining (OM), gender identification (GI), quality assessment (QA), complaint detection (CD), and component isolation (CI)

Literature	Tasks	Data			
		Email	Review	News	Forum
(*De Vel*, 2000)	AI	√			
(*De Vel et al.*, 2001)	AI	√			
(*Zheng et al.*, 2006)	AI				√
(*Abbasi and Chen*, 2008)	AI	√	√		√
(*Pang et al.*, 2002)	OM		√		
(*Wilson et al.*, 2005)	OM			√	
(*Corney et al.*, 2002)	GI	√			
(*Zhang et al.*, 2011)	GI				√
(*Weimer et al.*, 2007)	QA				√
(*Baldwin et al.*, 2007)	QA				√
(*Coussement and Vandenpoel*, 2008)	CD	√			
(*Abrahams et al.*, 2013, forthcoming)	CI				√

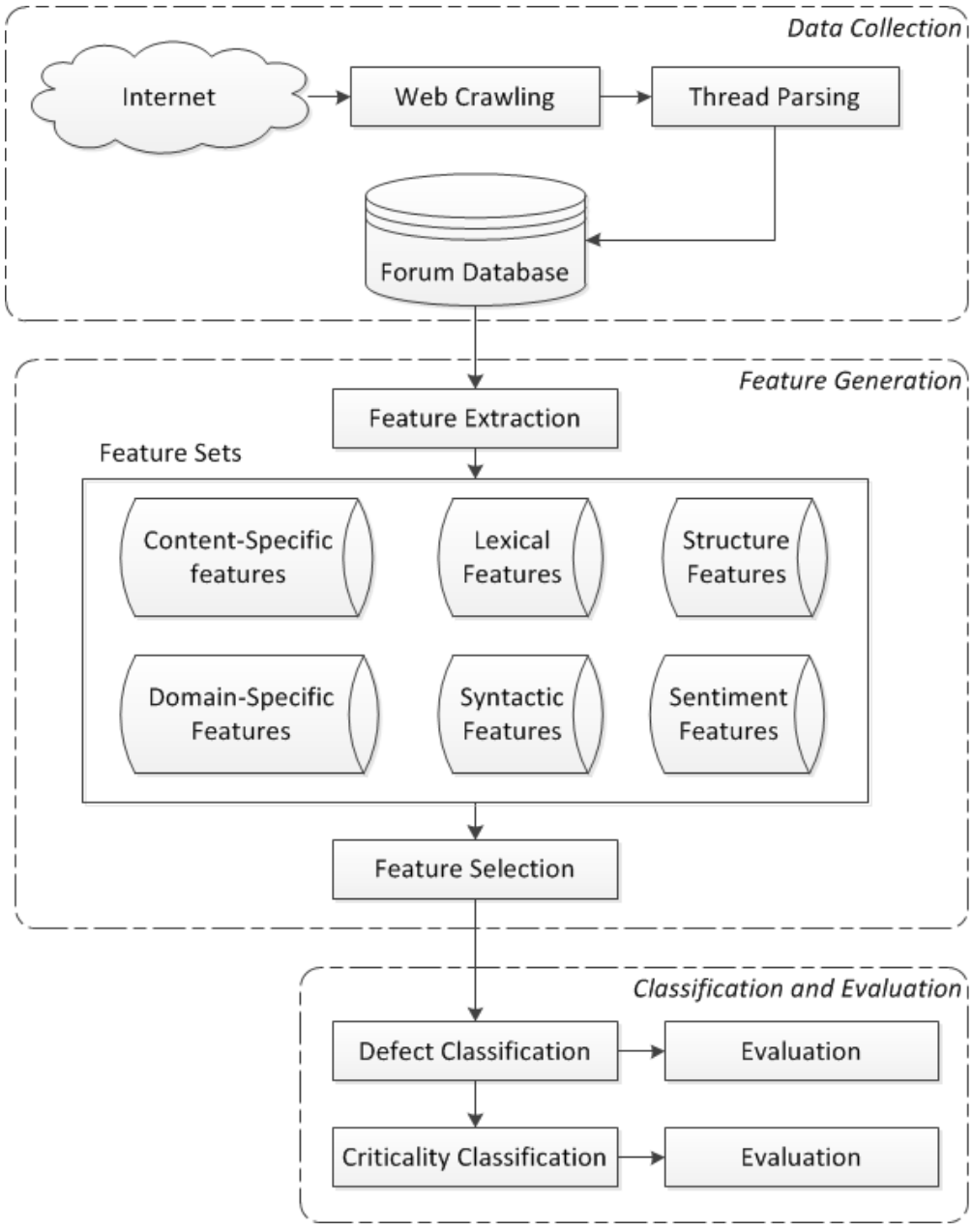

Figure 2.1: System architecture of product defect detection

content-free features. Content-specific features are essentially word and phrase features that are used to represent certain underlying topics. There are two variations of content-specific features, unigram and n-gram. Content-free features are features not related to document topics, including sentiment features, domain specific features, lexical features, syntactic features, and structure features. All these features are carefully selected and generate the final feature set for classification.

The last component is *classification and evaluation*, whereby different classification techniques are applied and evaluated based on a set of standard and well-defined performance metrics. The entire data set is partitioned into subsets to make sure there are sufficient data for training and, also, that the over-fitting problem is alleviated.

2.3.1 Classifiers

The classifiers used in the experiments are support vector machine (SVM) (*Gunn*, 1998; *Hearst et al.*, 1998), neural networks (NN)(*Bishop*, 1995; *Lippmann*, 1987; *Dagan et al.*, 1997), decision trees (J48)(*Breiman et al.*, 1984; *Quinlan*, 1993), and logistic regression (LR) (*Hosmer and Lemeshow*, 2000).

2.3.2 Features extraction and selection

Feature sets

Features are very important in supervised classification: good features can greatly improve performance. Multiple types of features are employed in our work to detect defects.

- Lexical features are features like the number of words and average word length. Such features proved to be very suitable for writing style classification and author identification. According to feature analysis results, it appears that defect threads contain relatively fewer words (444 vs. 536), but the difference in average thread length in defect and non-defect threads does not seem to be significant ($p = 0.48$).
- Syntactic features include part-of-speech distribution, punctuation distribution, question sentence rate, and others, which are expected to capture some of the patterns people use in describing a defect.
- The number of sentences per paragraph and the number of paragraphs per document are traditional structure features. In a Web forum, each thread contains one or more posts, and these posts are made by one or more users. Based on this information, some new structure features are proposed that include number of users, numbers of posts, etc.

- Content-specific features are really rich features and are widely used in text classification. They are simply terms or phrases extracted from plain texts, and they are the most intuitive features that people may think of. Content-specific features were extracted in three ways:
  - From the term index of forum threads, one can easily tell if a term appears in a thread or not. With such information and thread labels, a two-way contingency table of defect existence and term appearance was created and the feature ranking methods discussed in Section 2.2.3 were employed to calculate a feature importance score for each term present in the collected data. Words with top scores were then selected as content-specific features. Four feature ranking methods were adopted to conduct this analysis including IG, CS, DRC and RSV, and n-gram term features ($n \leq 3$) are added to ranking.
  - General Inquirer is a content analysis tool (*Stone et al.*, 1962, 1966) that maps a text document to a vector of counts based on a list of dictionary-supplied categories. There are in total 182 general inquirer categories, which combine Harvard IV-4 and Lasswell categories, and also include some new categories like positive, negative, pleasure, etc. In order to produce mapping with reasonable accuracy, the tool can map words with different forms to the same word root and meanwhile distinguish different senses of the same word.
  - Latent Dirichlet allocation (LDA) (*Blei et al.*, 2003; *Griffiths and Steyvers*, 2004) is a generative model, which can be used to find the underlying topics of documents. According to LDA, product defect threads can be modeled by a mixture of a set of topics and therefore these topics can be naturally used as content-specific features.
- It was mentioned in (*Abbasi et al.*, 2008) that sentiments could be potentially used in the analysis of inferior features about products that may concern customers and therefore could be good indicators of product defects. To investigate this using OpinionFinder (*Wiebe et al.*, 2005; *Wilson et al.*, 2005), the opinion sentences in each thread were extracted, and three categories of opinions were identified: subjective, positive, and negative. The ratios of sentences containing these categories of sentiments were calculated as features.
- Some components of products are more vulnerable and tend to have more problems than others. Therefore, product components could be good indicators of defects, which are arguably domain dependent. In our work, auto parts/components are used as domain specific features.

Feature levels

Distinct from plain text documents, which are unstructured, forum threads are organized in reply trees. Each forum thread can be structured as a tree, with vertices representing the posts and the direct edges representing the replying relationship between posts, pointing from replies to the quoted posts. Figure 2.2 illustrates a thread reply tree, where each post vertex is labeled with the order of its posting (P_n) and the community member who created it (u_m).

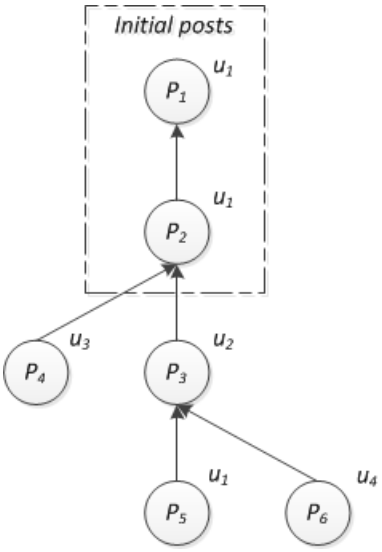

Figure 2.2: Thread reply tree

Given such a structure, information can be extracted at different granularities. Similar to (*Baldwin et al.*, 2007), two levels of information are identified, and features are extracted at both levels.

- Initial posts include the first post of the thread and the immediately proceeding posts from the same author based on the chronological order of the posts. The author of the first post starts the discussion by asking a question and may clarify or supplement the question in subsequent posts. The purpose of the initiating author is to seek helpful information, which is usually expressed directly and clearly. As the discussion continues, however, the topic may drift. People may start talking about something else not related to the initial post. For this reason, initial posts usually contain the clearest statement of the problem.
- All responses include all posts in a thread. All responses tell a complete story about the problem and may contain a solution or answer to the problem.

Feature normalization

Three different data sets were used in the experimental evaluation. Since these forums are independent of each other, they vary in the number of registered members, number of active users, and average thread length. Plus, the underlying distribution of defects is different. To minimize the impact of all these factors, the counting features are normalized using the z-score.

$$z = \frac{x - \mu}{\sigma}$$

where μ is the mean of feature values for all the threads in a particular forum and σ is the standard deviation.

A full list of features is presented in Table 2.4.

Table 2.4: Defect detection features. Two levels for some features: Initial posts (I) and all responses (A)

Features	
Lexical features and linguistic features	
$F11$	Total number of words (I and A)
$F12$	Total number of unique words (I and A)
Syntactic features	
$F21$	Percentage of punctuation marks (10 features)
$F22$	Percentage of part-of-speech (87 features)
$F23$	Percentage of question sentences (I and A)
$F24$	Percentage of exclamation sentences (I and A)
Structure features	
$F31$	Total number of sentences
$F32$	Total number of posts (Z-Score)
$F33$	Total number of users (Z-Score)
$F34$	Total number of views (Z-Score)
Content-specific features	
$F41$	General Inquirer categories (I and A, $2 \times 182 = 364$ features)
$F42$	LDA topic features (40 topic features)
$F43$	Salient keywords selected using feature ranking (200 features)
Sentiment features	
$F51$	Percentage of positive sentences
$F52$	Percentage of negative sentences
$F53$	Percentage of subjective sentences
Domain-specific features	
$F61$	Component category (13 features)

2.4 Experimental results

The experiment section is divided into four parts. First, the evaluation questions are presented. The second part describes the experiment data sets and the process of collecting data. Next, performance metrics are discussed. Finally, the results of defect detection and criticality classification are reported.

2.4.1 Evaluation questions

The algorithms for defect detection and criticality classification were evaluated separately, and the effectiveness of different features was also assessed. In brief, the experiments were designed to answer the following questions:

Question 1. *What features are effective in defect detection?*

Question 2. *What is the performance of automated defect detection?*

Question 3. *Can the same set of features be used to classify criticality levels?*

Question 4. *How well does the criticality classifier perform?*

2.4.2 Data sets

The study was conducted on data from the online forums of three top-selling car makers in the U.S. market: Honda, Toyota, and Chevrolet. We obtained the permission of the forum owners and crawled three online forums: Honda-Tech.com, ToyotaNation.com, and ChevroletForum.com in June 2010. Table 2.5 summarizes information about the three forums.

Table 2.5: Forum data sets

Forum	number of users	Number of sub-forums	Number of threads
Honda	201,975	34	1,316,881
Toyota	78,203	100	216,599
Chevrolet	18,498	48	26,936

The topics discussed in the forums can be very broad, ranging from general brand discussion to technical details, and defect-related threads only comprise a small fraction of all the forum threads. Therefore, it was impractical for us to go through all of the threads to identify those that relate to defects. In order to obtain a set of threads with a relatively balanced number of instances from different categories, we needed a strategy to filter most of the non-defect threads.

The National Highway Traffic Safety Administration (NHTSA) hosts a website that allows vehicle owners to file complaints about product defects. Government analysts then investigate these submitted complaints to determine whether they are safety defects or not and how to deal with them. We downloaded the 2010 snapshot of the vehicle safety complaint database and extracted the top 200 most frequent keywords appearing in defective component descriptions (stop words were removed). Moreover, we eliminated sub-forums that focused on racing, marketplace and car clubs and selected the top 1,500 threads containing the highest frequency of the keywords from each of the three forums.

We employed three independent motor vehicle domain experts to conduct the thread tagging. Two of them were graduate students, and the third one was a senior undergraduate student. All were automotive engineering majors from a nationally recognized vehicle systems and safety research group. The experts were required to read each forum thread assigned to them and make decisions based on the following four questions:

1. Does the thread discuss a product defect?
2. If yes, how critical is the defect?
3. What vehicle component category does this thread belong to?
4. What topic is discussed in the thread?

The input from these domain experts was collected, and sometimes discrepancies in expert judgments were resolved by majority voting. Gold set labels of defects and criticality levels were generated based on the tagging of the first two questions, and the last two labels were collected as data preparation for further research. The defect distribution can be found in Table 2.4.2.

Table 2.6: Defect distribution in the data sets

Forum	Defect		Non-defect	Total
	Safety	Performance	Other	
Honda	193	879	428	1,500
Toyota	408	355	737	1,500
Chevrolet	130	507	863	1,500
Total	731	1,741	2,028	4,500

To alleviate the problem of over-fitting on the training data set and to get a generalizable classification model, we followed one of the experiment designs described in (*Alpaydin*, 2010) and randomly partitioned the entire data set into two subsets: a train-and-validation set (70%) and a test set (30%): the train-and-validation set would be used to find optimal feature sets and tune model parameters, while the test set is a hold out set with a reasonably large amount of data to evaluate the model. The flowchart is illustrated in Figure 2.3.

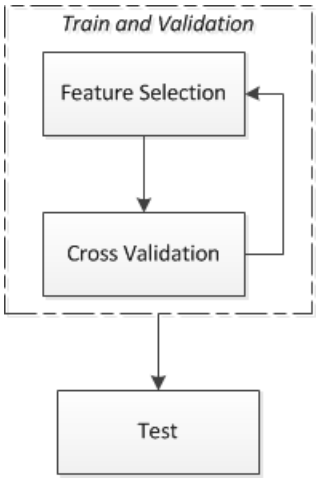

Figure 2.3: Defect detection system evaluation flowchart

At the beginning, a full set of features was extracted, and an initial set of features was selected using information gain, a correlation metric, on the train-and-validation set. Then, the wrapper method was adopted to combine feature selection with selected classifiers, and the performance of features was gauged based on a 10-fold cross-validation of results. The 10-fold cross-validation was carried out on account of the benefits of allowing us to have sufficient data for training while avoiding the bias introduced by data set partition. According to the averaged performance, the feature set was adjusted and a new feature selection was generated. Based on these new features, another round of cross-validation was carried out. This process was repeated until best performance was achieved and there was no room to improve. Finally, the entire train-and-validation data set was used to train a model based on the optimal feature set, and performance was assessed on the test data set. All reported performance numbers are from the test data set.

2.4.3 Experimental metrics

Performance of the defect classification is evaluated using the following measurements: precision (P), recall (R) and f-measure (F_1 and F_2). The first two metrics and F_1 have been extensively used in the areas of information retrieval, document classification and query classification (*Beitzel*, 2006). The last one is a variant of F_1 (*Rijsbergen*, 1979), which doubles the weight of recall in the precision-recall combination and is more focused on recall. F_2 is introduced because not missing a single defect is really important, although there could be some loss in precision. The binary classification confusion table is given below.

where A denotes the number of instances with labels equal to 1 that are predicted as 1, B denotes the number of instances with labels equal to 1 that are predicted as 0, C denotes the number of instances with labels equal to 0 that are predicted as 1, and D denotes the

Table 2.7: Two-way confusion table

		Predicted class	
		1	0
Actual class	1	A	B
	0	C	D

number of instances with labels equal to 0 that are predicted as 0. Given the two-way confusion matrix, the metrics can be defined as so:

- Accuracy
$$\text{accuracy} = \frac{A + D}{A + B + C + D}$$
- Precision
$$\text{precision} = \frac{A}{A + C}$$
- Recall
$$\text{recall} = \frac{A}{A + B}$$
- F-measure
$$F_\beta = \frac{(1 + \beta^2)P \cdot R}{(\beta^2)P + R}$$
If precision and recall are equally important ($\beta = 1$), we have F_1 as
$$F_1 = \frac{2 \cdot P \cdot R}{P + R}$$
If recall is twice as important as precision ($\beta = 2$), we have F_2 as
$$F_2 = \frac{5 \cdot P \cdot R}{4 \cdot P + R}$$
- Chi-square The chi-square statistic (*Alpaydin*, 2010) is defined as
$$\chi_1^2 = \frac{|(e_{01} - e_{10}| - 1)^2}{e_{01} + e_{10}}$$
where e_{01} means the number of instances that classifier 2 predicts correctly but classifier 1 fails while e_{10} means the opposite. Chi-square was used to measure the significance.

2.4.4 Experiment results

Defect detection

Among the literatures cited in Section 2.2, (*Weimer et al.*, 2007) is closest to our work: 1) the data sets are both forum data and therefore most of the features could be potentially reused ; 2) both tasks require deep analysis into the thread content. Based on the above two points, the approach in (*Weimer et al.*, 2007) was selected as the baseline method, where a SVM classifier was employed with three types of features, $F1$, $F2$ and $F3$. To measure the performance more precisely, some positive examples were removed from the test set to balance the number of defects and non-defects. The results of the baseline method and other classification methods are shown in Table 2.8.

Table 2.8: Results for test set (baseline results included) with feature sets: $F1$, $F2$ and $F3$

Method	Accuracy	Precision	Recall	F_1	F_2
J48	0.587	0.58	0.633	0.605	0.622
Logistic	0.591	0.568	0.765	0.652	0.715
NN	0.570	0.543	0.878	0.671	0.782
SVM (Baseline)	**0.567**	**0.544**	**0.832**	**0.657**	**0.752**

According to the results, the accuracy and F_1 for the baseline method are 0.567 and 0.657, respectively. Logistic regression and neural networks produced comparable results, with both of them performing slightly better than baseline. The former has higher overall accuracy but F_1 is a little bit lower. Although all the methods are better than flipping a coin, none of them is practically applicable considering the lower than 60% accuracy. More powerful features are needed.

General Inquirer categories (*Stone et al.*, 1962, 1966) were developed based on content analysis and were adopted here to extract content-specific features. The new features were combined with existing features from $F1$, $F2$ and $F3$ to form a new feature set. The information gain between each feature column and the defect label was calculated and features with low correlation scores were removed from the feature list. Greedy stepwise method was used together with each of the proposed classification models to search for a subset of features that can achieve the best cross-evaluation results.

Table 2.9: Results for defect detection on test set with feature sets: $F1$, $F2$, $F3$, and $F41$ (General Inquirer category features)

Method	Accuracy	Precision	Recall	F_1	F_2
J48	0.537	0.532	0.629	0.576	0.607
Logistic	**0.606**	**0.585**	**0.733**	**0.651**	**0.698**
NN	0.571	0.552	0.754	0.638	0.703
SVM	0.584	0.554	0.873	0.678	0.783

Table 2.9 shows the classification results with additional General Inquirer category features on the test data set. SVM performed the best in terms of F_1 while logistic regression got higher than 60% accuracy. With the addition of General Inquirer category features, the performance was improved about 2%.

Another attempt at new features is to incorporate new content-specific features, latent Dirichlet allocation (LDA) topic features (*Blei et al.*, 2003; *Griffiths and Steyvers*, 2004), into the feature set for defect detection. The data from a sub-forum of Honda-tech, Honda Accord (1990-2002), were used to train the LDA model and then infer the topic distribution over all threads in the defect detection data set. Stanford topic modeling toolbox was used in our study for LDA analysis (*Ramage and Rosen*, 2009). LDA is a parametric model and the parameter of topic number needs to be determined first.

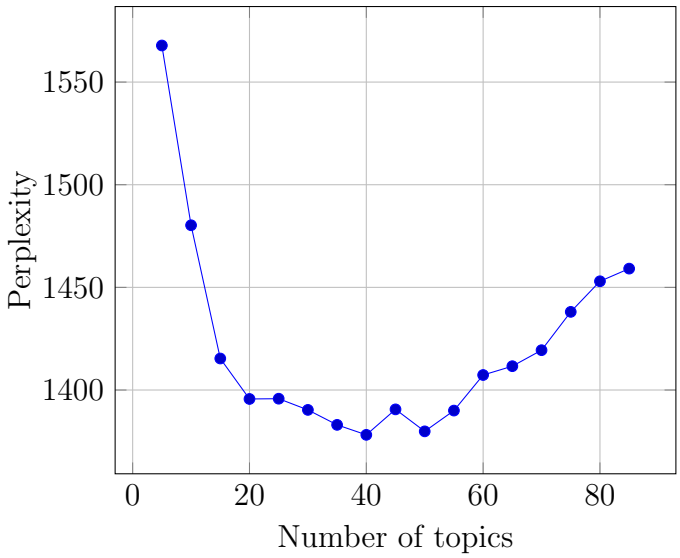

Figure 2.4: Perplexity over different numbers of topics

Figure 2.4 illustrates the perplexity curve on a holdout data set over a varying number of topics. It appears that when the number of topics was equal to 40, perplexity reached the minimum. Taking 40 as the parameter value of the number of topics, a LDA model was learned, and 40 features were generated corresponding to the LDA topics and added to the feature set.

With the addition of LDA features, after feature selection, the performance of SVM improved significantly, with a 12.5% gain in accuracy and a 4.7% gain in F_1. Among the top 10 features carrying the highest weights in the SVM model, four of them are LDA topic features, as listed in Table 2.11.

Next, content-specific term features were introduced. Different from published studies, where term features were manually selected, we first used a heuristic method to automatically find content-specific term features. Term ranking methods, which were previously widely used in

Table 2.10: Results for defect detection on the test set with feature sets: $F1$, $F2$, $F3$, $F41$ and $F42$ (LDA topic features)

Method	Accuracy	Precision	Recall	F_1	F_2
J48	0.554	0.546	0.652	0.594	0.628
Logistic	0.637	0.617	0.724	0.666	0.700
NN	0.597	0.571	0.787	0.662	0.732
SVM	**0.638**	**0.605**	**0.798**	**0.688**	**0.750**

Table 2.11: Top LDA features

Topic number	Representative keywords
topic 5	belt, valve, pump, pulley, crank
topic 8	turbo, boost, stock, run, kit
topic 18	sound, noise, drive, hear, turn
topic 22	bolt, remove, nut, take, tool

text classification and information retrieval, were adopted to select prominent term features. These methods are information gain (IG), chi-square (CS), document and relevance correlation (DRC), and Robertson's selection value (RSV) (*Fan et al.*, 2005; *Yang and Pedersen*, 1997). To evaluate the effectiveness of these methods and determine the number of term features needed, two text classification methods were employed to combine with the term ranking methods. The 10-fold cross-validation results on the train-and-validation set are illustrated in Figure 2.5.

According to the results, term selection methods CS, IG, and RSV were performing equally well. Another finding was that SVM is generally better than naive Bayes in terms of F_1. The corresponding numbers on the test set are shown in Table 2.12, where the number of terms was determined according to the best performance on the train-and-validation set.

Table 2.12: Defect detection results on the test set using unigram term features only

Method	Number of terms	Accuracy	Precision	Recall	F_1	F_2
SVM+CS	400	0.6508	0.6267	0.7460	0.6812	0.7187
NB+CS	200	0.5992	0.5846	0.6857	0.6311	0.6628
SVM+DRC	300	0.6238	0.6034	0.7222	0.6575	0.6949
NB+DRC	800	0.5849	0.5636	0.7524	0.6445	0.7051
SVM+IG	300	0.6389	0.6061	0.7937	0.6873	0.7474
NB+IG	200	0.5968	0.5838	0.6746	0.6259	0.6542
SVM+RSV	100	0.6341	0.5986	0.8143	0.6900	0.7595
NB+RSV	300	0.5992	0.5839	0.6905	0.6327	0.6662

As we can see from Table 2.12, the term only features (RSV) are very powerful; when combined with the SVM classifier, it produced results as high as 0.69 F_1 and 0.634 accuracy. These features should definitely be included in the feature set.

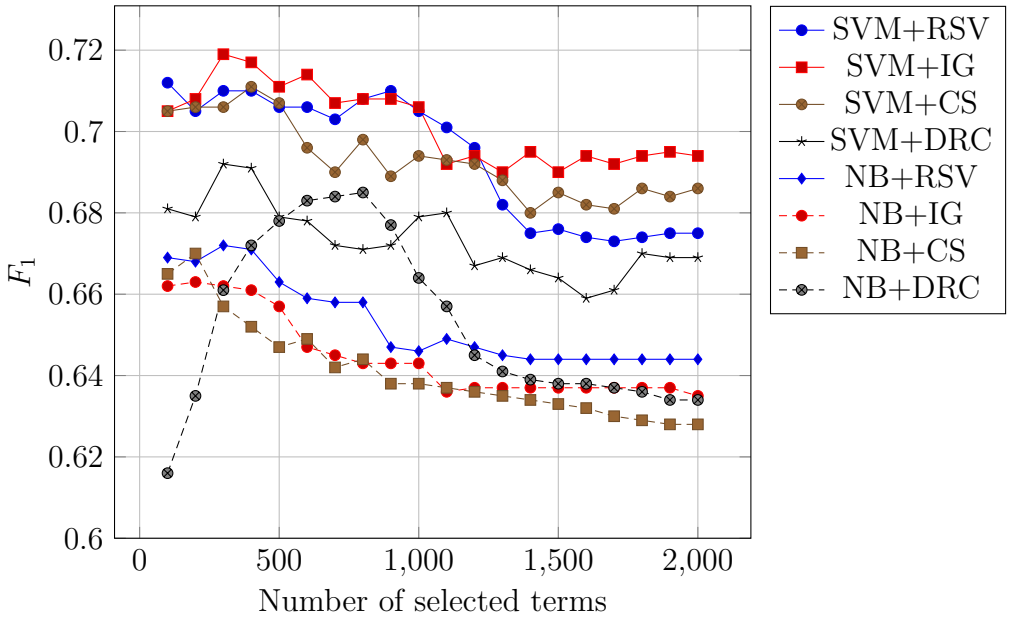

Figure 2.5: Cross-validation defect detection results over varying numbers of terms

Single terms may not sufficiently express the post authors' opinions and effectively distinguish defects from normal posts (*Abbasi and Chen*, 2008). Next, we looked into one-to-three word breaking and explored the classification performance at a different granularity. The cross-validation results and test results are given in Figure 2.6 and Table 2.13 separately.

Table 2.13: Defect detection results on the test set using n-gram term features only

Method	Number of terms	Accuracy	Precision	Recall	F_1	F_2
SVM+CS	100	0.6413	0.6057	0.8095	0.6929	0.7585
NB+CS	100	0.6135	0.5970	0.6984	0.6437	0.6755
SVM+DRC	600	0.6365	0.6211	0.7000	0.6582	0.6827
NB+DRC	1200	0.5905	0.5675	0.7603	0.6499	0.7120
SVM+IG	100	0.6111	0.5749	0.8524	0.6867	0.7774
NB+IG	1000	0.6095	0.6006	0.6540	0.6261	0.6425
SVM+RSV	100	0.6310	0.5904	0.8556	0.6986	0.7850
NB+RSV	300	0.6071	0.5913	0.6937	0.6384	0.6705

According to the results, there is only marginal improvement using n-gram compared to the unigram terms. From the cross-validation results of both unigram and n-gram terms, the RSV method appears to be very effective in selecting good terms for defect identification and at the same time the the RSV F_1 curve looks relatively flat and stable. The top 100 n-gram terms were selected using the RSV method and incorporated into the feature selection for defect detection.

Table 2.14 shows the classification results on the test set using feature sets: F_1, F_2, F_3, and

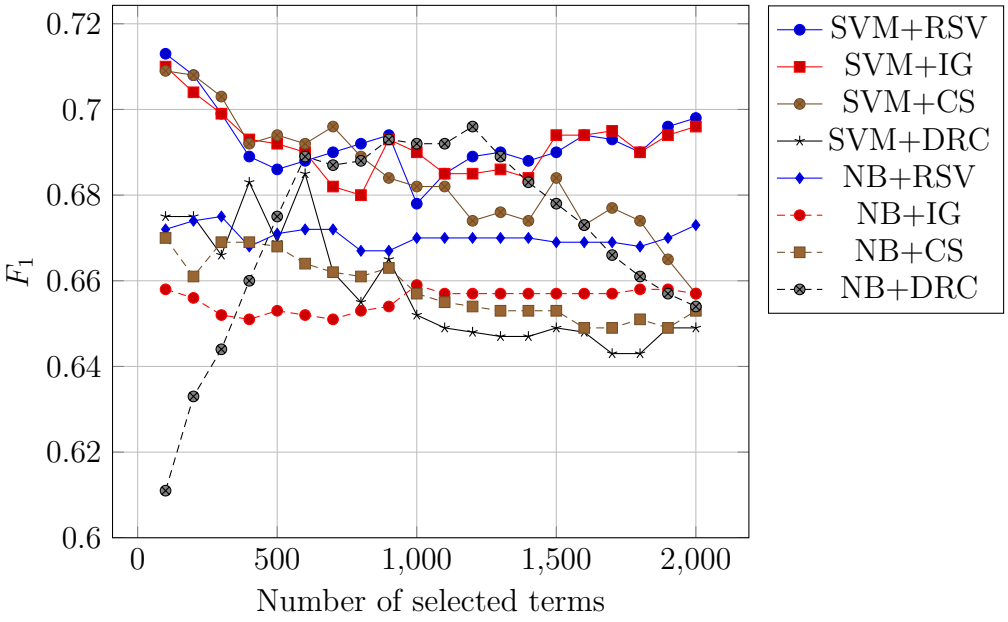

Figure 2.6: Cross-validation defect detection results over varying numbers of n-gram terms

Table 2.14: Results for defect detection on the test set with feature sets: $F1$, $F2$, $F3$, and $F4$

Method	Accuracy	Precision	Recall	F_1	F_2
J48	0.587	0.569	0.721	0.636	0.684
Logistic	0.643	0.617	0.759	0.68	0.726
NN	0.582	0.588	0.548	0.567	0.556
SVM	**0.653**	**0.616**	**0.817**	**0.703**	**0.767**

F_4. The SVM classifier produced the best results we can achieve so far, with up to **0.653** accuracy and **0.703** F_1, and it has better recall than precision, which is exactly what we need. The chi-square is 37 (> 3.84), which means this method is significantly better than the baseline method.

To investigate the relationship between sentiments and defects, sentiment analysis was performed to extract users' subjective opinions on vehicles from thread content. Specifically, positive/negative/subjective sentence ratios were calculated and added to the feature set.

Table 2.15: Results for defect detection on test set with feature sets: $F1$, $F2$, $F3$, $F4$ and $F5$

Method	Accuracy	Precision	Recall	F_1	F_2
J48	0.575	0.555	0.756	0.64	0.705
Logistic	0.646	0.622	0.746	0.678	0.717
NN	0.633	0.597	0.819	0.691	0.762
SVM	**0.672**	**0.638**	**0.797**	**0.709**	**0.759**

Table 2.15 shows the classification results using feature sets: $F1$, $F2$, $F3$, $F4$ and $F5$. According to the results, there is no significant gain compared to the experiment results without sentiment features (chi-square equal to 0.289). This could either be because user sentiments have no contribution to separating defects from non-defects or because they are already covered by the other features.

While working on the defect tagging, we noticed that some components are shaky and tend to have more problems than other parts. Therefore, vehicle components as features were considered to have high potential to improve classification performance. As the defect labels were collected, the judges were also required to tag the vehicle components that each thread belonged to. Table 2.16 shows some statistics of the component labels we collected.

The idea is to use these component labels as defect detection features. However, it is not realistic to require all the forums to provide component tagging information and moreover, it is very expensive to collect such labels. An alternative solution is to predict the component tags in an automated way; however, the tags we collected, which are needed to train the automated model, are not sufficient and this will greatly impact the accuracy of classification. Fortunately, besides the model-year sub-forums, the Honda-tech data set also organizes threads by components. Both types of sub-forums coexist, and one has the freedom to select the sub-forum to start a discussion. The component sub-forums as labels can be naturally used to train the component isolation model. Ten key component sub-forums were selected, as listed in Table 2.17.

Actually, the thread components are very sparse: some threads were not in any of these component sub-forums (as mentioned before, there also exist sub-forums organized by model-years), and some threads could be tied to multiple components, e.g., threads about brake lights could be related to both brakes or lighting, although they came from one specific

Table 2.16: Component tags

Component	Number of threads	Percentage
Air conditioning	261	5.80%
Airbag	116	2.58%
Braking	286	6.36%
Electrical system	1,625	36.11%
Engine	1,815	40.33%
Lights	122	2.71%
Seat belts	61	1.36%
Steering	119	2.64%
Structure and body	319	7.09%
Transmission	663	14.73%
Visibility	96	2.13%
Wheels and tires	8	0.18%
Other	321	7.13%
Suspension	68	1.51%
Acoustics	72	1.60%

Table 2.17: 10 sub-forums from Honda-tech forum

ID	Forum name	Thread counts
19	Road racing/autocross	30,863
27	Audio/security/video	24,124
53	Welding/fabrication	9,117
54	Suspension	16,395
56	Wheel and tire	11,572
84	Paint and body	4,721
107	Lighting	1,872
124	Engine management and tuning	2,090
127	Transmission drivetrain	1,917
128	Brakes	620

sub-forum. Automated classification also can solve this data sparsity problem.

If we consider sub-forums as component labels, we could train a multi-label classification model to categorize the threads into components and use the predicted components as features. First, the multi-label classification problem was converted into 10 binary classification problems. We sampled up to 5,000 threads from each sub-forum as positive examples and sampled another 5,000 threads or the equivalent number of threads from all other sub-forums as negative examples to train 10 classifiers. Then the classifiers were applied to the 4,500 threads in our defect detection data set. The results are shown below.

Table 2.18: 5-fold cross-validation classification using component category human judgments as labels

Component category	Precision	Recall	F_1
Acoustics	0.0481	0.6923	0.0900
Air conditioning	0.4388	0.9149	0.5931
Airbag	0.2414	0.8750	0.3784
Braking	0.4462	0.9206	0.6010
Electrical system	0.7089	0.8018	0.7525
Engine	0.7919	0.8371	0.8139
Lights	0.1429	0.8621	0.2451
Other	0.1319	0.6441	0.2190
Seat belts	0.1486	1.0000	0.2588
Steering	0.2424	0.9600	0.3871
Structure and body	0.3293	0.8710	0.4779
Suspension	0.1250	0.8571	0.2182
Transmission	0.6199	0.8896	0.7307
Visibility	0.1618	1.0000	0.2785
Wheels and tires	0.1193	0.9286	0.2114

Table 2.19: 5-fold cross-validation classification using Honda sub-forum as labels

Component(sub-forum)	Precision	Recall	F_1
Road racing/autocross	0.9588	0.9387	0.9487
Audio/security/video	0.9599	0.9489	0.9544
Welding/fabrication	0.9572	0.9430	0.9501
Suspension	0.9600	0.9508	0.9554
Wheel and tire	0.9822	0.9733	0.9777
Paint and body	0.9333	0.9270	0.9302
Lighting	0.9347	0.9068	0.9205
Engine management and tuning	0.9154	0.8530	0.8831
Transmission drivetrain	0.9731	0.9612	0.9671
Brakes	0.9200	0.9411	0.9305

A 5-fold cross-validation was performed to understand the performance of this multi-label

classifier. Table 2.19 demonstrates the results of component classification and the average F_1 is as high as 0.94, which means component classification models trained based on sub-forum labels could achieve very high accuracy and are more reliable than classifiers trained based on limited human labels. As a result, by combining the two sets of predicted components from both sources, we obtained 13 component features.

Table 2.20: Results on the test set with feature sets: $F1$, $F2$, $F3$, $F4$, $F5$ and $F6$

Method	Accuracy	Precision	Recall	F_1	F_2
J48	0.569	0.543	0.867	0.668	0.775
Logistic	0.646	0.623	0.741	0.677	0.714
NN	0.579	0.549	0.892	0.68	0.793
SVM	0.648	0.613	0.8	0.694	0.754

The results shown in Table 2.20, with vehicle component features included, are comparable to those in Table 2.15 (chi-square = 1.1) and not very promising. It is likely that the characteristics of components overlapped with the other features, e.g., term features. In conclusion, the best method of defect defection is SVM on feature sets $F1$, $F2$, $F3$, $F4$ and $F5$ with accuracy equal to 0.672 and F_1 equal to 0.709.

Criticality classification

In the following part, we describe the experimental results of safety vs. non-safety defects. To be consistent with previous experiments, both the train-and-validation set and the test data set have the same sets of threads.

Table 2.21: Safety defect distribution

Partition	Safety	Non-safety defect	Safety defect ratio
Train-and-validation	535	1,217	30.5%
Test	196	524	27.2%

Table 2.21 shows the distribution of safety defects in two data partitions. Since there are many more performance defects than safety defects, to keep the balance of positive and negative examples, and at the same time avoid the bias of performance evaluation, safety defects were replicated in the train-and-validation set and the extra performance defects were taken out of the test set. Table 2.22 gives the new distribution.

Table 2.22: Balanced safety defect distribution

Partition	Safety	Non-safety defect	Safety defect ratio
Train-and-validation	1,070	1,217	46.8%
Test	196	196	50%

Most of the features used in defect identification can also be used for classifying criticality, except content-specific term features, which require computation of defect criticality and term occurrence correlation. The same feature ranking methods were applied to find the most prominent term features.

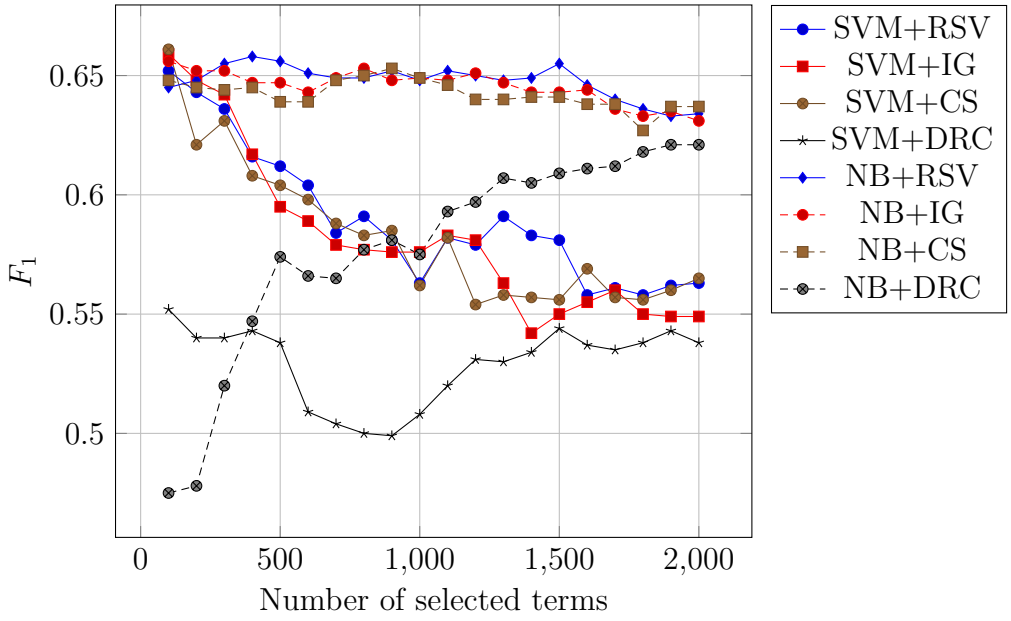

Figure 2.7: Cross-validation F_1 results over different numbers of terms

Figure 2.7 shows the cross-validation F_1 results over a varying number of selected terms. The RSV, CS, and IG methods performed equally well with both classifiers when the number of terms equaled 100. In cross-validation, the training step was performed on a balanced data set, but the validation step was performed on the remaining unbalanced data set, where performance defects dominated. That accounts for why the F_1 is a little bit lower than the real situation.

Table 2.23: Safety classification performance using unigram term features only

Method	No. of terms	Accuracy	Precision	Recall	F_1	F_2
SVM+CS	100	0.7679	0.8302	0.6735	0.7437	0.6999
NB+CS	900	0.7526	0.7463	0.7653	0.7557	0.7614
SVM+DRC	100	0.6811	0.6961	0.6429	0.6684	0.6528
NB+DRC	2000	0.7245	0.7245	0.7245	0.7245	0.7245
SVM+IG	100	0.7577	0.8061	0.6786	0.7368	0.7007
NB+IG	100	0.7219	0.7062	0.7602	0.7322	0.7487
SVM+RSV	100	0.7653	0.7955	0.7143	0.7527	0.7292
NB+RSV	400	0.7296	0.7296	0.7296	0.7296	0.7296

Table 2.23 shows the performance of safety classification on the test data set. As we can see

from the table, using term features alone can produce very promising results, with F_1 equal to 0.75 and accuracy equal to 0.75. Below, n-gram term selection results are provided.

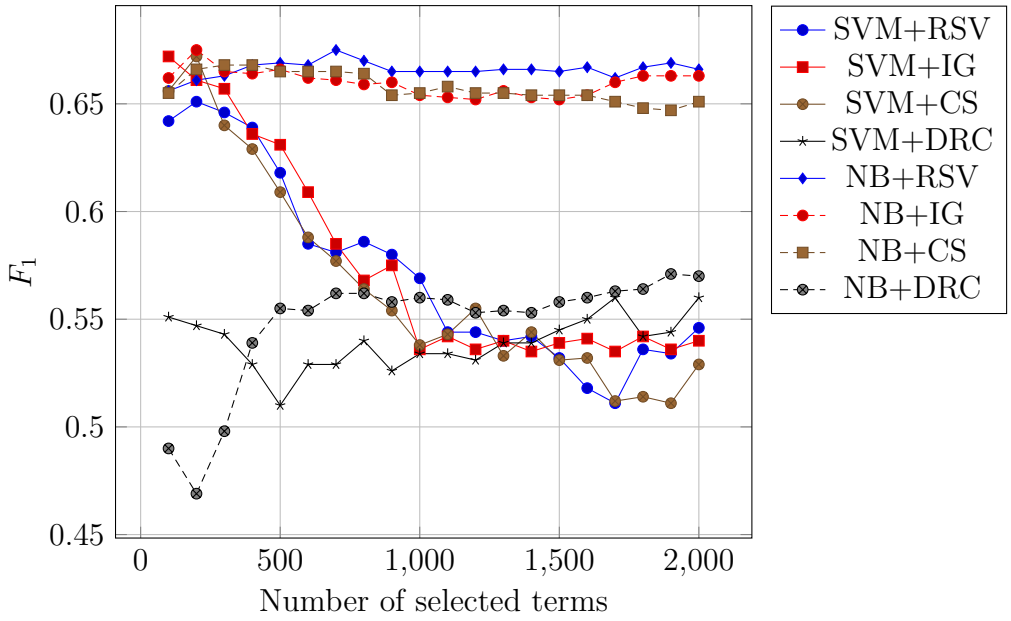

Figure 2.8: Cross-validation F_1 results over different numbers of n-gram terms

Method	No. of terms	Accuracy	Precision	Recall	F_1	F_2
SVM+CS	200	0.7908	0.8851	0.6684	0.7616	0.7028
NB+CS	300	0.7602	0.7476	0.7857	0.7662	0.7778
SVM+DRC	2000	0.7168	0.7852	0.5969	0.6783	0.6270
NB+DRC	1900	0.6735	0.6382	0.8010	0.7104	0.7621
SVM+IG	100	0.7730	0.8497	0.6633	0.7450	0.6937
NB+IG	200	0.7449	0.7286	0.7806	0.7537	0.7696
SVM+RSV	200	0.7704	0.8630	0.6429	0.7368	0.6774
NB+RSV	700	0.7551	0.7404	0.7857	0.7624	0.7762

Table 2.24: Safety classification performance using n-gram features

N-gram did not bring us too much gain based on the evaluation results, but considering the fact that n-gram terms include all unigram terms already, they were used in the following experiments to combine with other features for classification. Since the CS term selection method with both SVM and NB has in general better performance, the top 200 terms selected using the CS method were include in the final feature set.

Table 2.25 demonstrates the classification results of safety defects vs. performance defects. The logistic regression method performs the best against all the other methods, with accuracy equal to 0.79 and F_1 equal to 0.78. The results are also better than term features-only results,

Table 2.25: Safety classification results on the test set with full feature set

Method	Accuracy	Precision	Recall	F_1	F_2
J48	0.706	0.758	0.607	0.674	0.632
Logistic	0.79	0.824	0.74	0.78	0.755
NN	0.711	0.786	0.582	0.669	0.613
SVM	0.783	0.825	0.719	0.768	0.738

which means there exist some effective non-term features. Table 2.26 lists the top 10 features with the highest weights in the logistic regression method.

Table 2.26: Top 10 features in logistic regression

Feature	Description
nrs	Proper nouns, e.g., region or newspaper
door stuck	n-gram
have drill	n-gram
vss vehicl speed	n-gram
vss vehicl	n-gram
my sr	n-gram
sr light	n-gram
airbag system	n-gram
topic23	LDA topic (brake/fluid/cylind/abs/pedal)

2.5 Conclusion

In this chapter, we proposed a framework based on supervised classification to detect the product defects and classify criticality levels of defects in the context of online forums. To evaluate the effectiveness of the framework, experiments were conducted on three different forum data sets, and the results showed that SVM and logistic regression outperformed the other two classification algorithms, with performances that were comparable and effective. Regarding feature types, content-specific features were shown to be the most prominent features.

Looking forward, the current performance is not quite satisfactory, and we will continue to improve the accuracy of defect detection. Moreover, we are collecting labels from the iPhone forum, which belongs to a completely different industry. We would like to see how the model performs on a different data set, whether there is any overlapping predominant feature terms in the vocabulary of the two, and what features remain effective across industries.

Chapter 3

Product Feature Extraction

Nowadays, online communities play an important role in many people's daily lives. To monitor and understand customer behavior and preferences, analysts have to deal with large collections of forum threads, and existing routing/filtration mechanisms such as defect thread detection are less than optimal ways of putting these data to good use. Summaries like topical key phrases or product features can substantially reduce the workload of analysis while lowering costs and improving efficiency. In this chapter, we present a machine learning method to automatically extract product features from forum threads and group similar feature terms into feature clusters. The experimental results of both extraction and clustering are very promising.

3.1 Introduction

With the rapid growth of online communities, online data have expanded exponentially, and there is growing interest in analyzing these data for business intelligence. However, the sheer volume of forum threads makes it very difficult for analysts to manually process them and extract much useful information. There is thus a high demand for a technological means of automatically summarizing massive forum threads using extracted topic keywords or product features.

Typically, an online forum is organized in a tree-like structure: one forum contains of a set of sub-forums that focus on one or more related topics; each of these sub-forums contains a series of threads in reverse chronological order with the latest threads listed at the top. For online customer forums specifically, the sub-forums may be devoted to particular product models, years of manufacture, or product categories, if there is more than one category of products. This forum structure is straightforward, easy to understand, and conducive to information browsing. Still, though, it leaves room for improvement. Most of the time, a customer's needs are specific and directed, but forum topics tend to be broad. For instance,

beyond models and years, a Honda customer may have a special interest in one transmission or acoustic system, and it may be difficult and time consuming for that consumer to locate the relevant information within the sub-forum structure. From the customers' perspective, an automated summarization system could greatly improve the efficiency and success of information browsing.

In addition to summarization, keywords extracted from documents are also very useful for information retrieval (*Jones and Staveley*, 1999) and document classification and clustering (*Han et al.*, 2007).

To get a sense of how information browsing works, consider the following example of a post thread on the topic of transmissions.

> **title**: *97* ***accord*** *V6* ***tranny problem****, help!!!*
> **content**: *The transmission shifts very strongly. Each* ***shift*** *is felt and heard. There is a strong change in the sound of the engine and the van jerks. This is true going up or down. The dealer says that this is normal. The explanation that I was given by the service technician is that this* ***transmission*** *locks up on each gear for better performance. This is the reason for the jerk. He also said that making a smooth* ***shift*** *puts a lot of strain on the* ***transmission*** *and the converter. I am not convinced, but I think that the jerky shifts are putting a lot of strain on the engine and the* ***power train****. I do know that they are putting a lot of stain on the passengers. Heck, I was able to get smoother* ***shifts*** *with my Honda, and I am not very good with manual shifts. Can anyone tell me whether this is normal?*

The above post is from a Honda forum (honda-tech.com) with auto features and other key phrases highlighted in bold. As shown in the post, product features are mostly nouns or noun phrases, which the discussion centers on. With a glance at the highlighted words, we can quickly grasp the main idea of the post: the user is seeking help for an automatic transmission problem with a Honda Accord V6 model.

In (*Hu and Liu*, 2004a), product features were extracted in the form of key noun phrases explicitly present in the text to summarize customer reviews. Similarly, we define the definition of a product feature as follows:

Definition 5. *A* ***product feature*** *is a feature or a component of a product or an important aspect or topic related to a product, which consists of one or more words and is referred to as f. In general, a large variety of product-related words or phrases can be regarded as product features, such as product components, features of product components and other related concepts.*

Based on this definition, a supervised classification method is proposed to automatically extract product features from forum threads. However, given the large volumes and great

expansion rate of forum data, single-thread summarization is still not enough. One may be more interested in aggregated information, e.g., which product feature is lately attracting the most attention from customers around the world. Furthermore, to express the same idea, different people may use different terms, so there is a need to build underlying conceptual connections among terms. To this end, beyond product feature extraction from single threads, we seek to group product features extracted from each thread into clusters and produce a summary of a group of threads as a whole. Following is the definition of a product feature cluster.

Definition 6. *A* ***product feature cluster*** *is a group of product features that are highly coherent and refer to the same or nearly the same aspect of a product. A cluster can be denoted as* $C = \{f_1, f_2, ..., f_l\}$*, where* f_i *is a product feature in the cluster. As an example, the two auto features "airbag" and "SRS light" are both related to a vehicle's safety system, and should be grouped together into one product feature cluster.*

This chapter investigates a thread summarization method in the context of online product forums and presents machine learning algorithms to summarize forum threads at two different levels of granularity. Experimental results from testing these methods exhibit that both are highly effective.

To avoid confusion, in the remainder of this chapter, features refer specifically to product features and attributes refer to the features of product features that are used in classification or clustering to characterize a product feature and separate it from others.

3.2 Literature review

Two tasks are covered in this chapter: product feature extraction and product feature clustering. The first task targets at a challenging problem and two areas of knowledge are required in order to solve it: key phrase-based document-level summarization and product feature extraction from online data. Previous studies in both areas are reviewed in the remainder of this section. Moreover, the last task focuses on a key phrase clustering problem and related work is query clustering, which is also discussed.

3.2.1 Keyword extraction

Key phrase extraction is an important branch of automatic text summarization. As discussed in (*Das and Martins*, 2007), automatic text summarization is the generation of a short text that conveys the important information of an original text or texts using computer technologies. The summaries can be extracted from either a single document or multiple documents, and the forms these extracts take can be paragraphs, sentences or phrases. Key phrase extraction from a forum thread is the focus of this chapter. Early work in this area

did a lot of exploration into the factors that measure the significance of salient parts of documents, and a rich set of features were then developed and widely used in practice. The attributes included word frequency, position in text, cue words, skeleton of document, inverse document frequency, and others (*Das and Martins*, 2007).

In designing an approach to summarizing product issues, we chose some attributes from the above list and developed some new attributes based on the special properties of online forum data sets.

After machine learning techniques were developed in 1990s, a series of seminal publications emerged that applied statistical methods to automatic text summarization, including Naive Bayes (*Kupiec et al.*, 1995; *Okurowski et al.*, 1999; *Witten et al.*, 1999), decision trees (*Lin*, 1999), hidden markov models (HMM) (*Conroy and O'leary*, 2001) and most recently learning to rank (*Jiang et al.*, 2009).

All the automatic text summarization research cited above focused on news articles or technical documents, which are usually well drafted by their authors and are different from customer feedback generated in the relaxed style common on the Web.

3.2.2 Product feature extraction

In last decade, many studies were conducted in mining customer opinions and extracting product features in customer reviews (*Hu and Liu*, 2004a,b; *Blair-Goldensohn et al.*, 2008; *Popescu and Etzioni*, 2005; *Titov and McDonald*, 2008; *Gamon et al.*, 2005). Aware of the rapid development of e-commerce and the importance of research in this area, Hu et al. (*Hu and Liu*, 2004a,b) initially proposed the idea of customer review summarization. Their work was different from traditional text summarization in that they were only interested in product features and customer opinions associated with them. So instead of key sentences, product features were extracted as summaries of product reviews. As the first step, they employed part-of-speech tagging (POS) to parse the reviews and identify nouns or noun phrases appearing in reviews as candidate features. Then, infrequent phrases with a frequency of less than 1% were removed from the original list. Finally, they pruned redundant and meaningless candidates and obtained product features using some simple rules. Results showed that their approach was effective in review summarization. Researchers in (*Popescu and Etzioni*, 2005) developed a system called OPINE that summarized product features in a different way. A set of seed products were first detected and the OPINE system recursively identified product features through some predefined relations and pre-extracted patterns. Sasha et al. presented an automated summarization system focused on local services in (*Blair-Goldensohn et al.*, 2008). The novel aspect of their model was the integration of user-provided labels and domain-specific information, and their results showed that quality was increased. The only data used by these studies were customer reviews. Although the possibility of applying similar models to newsgroups was mentioned in (*Popescu and Etzioni*, 2005), to the best of our knowledge, very little research has been carried out in the context

of online forums.

Table 3.1 lists some of the most prominent attributes that have been used in previous literature.

3.2.3 Product feature clustering

Several techniques that are related and could be potentially applied to group product features are query clustering (*Wen and Zhang*, 2002; *Fu et al.*, 2004) and topic models (*Hofmann*, 1999; *Blei et al.*, 2003; *Griffiths and Steyvers*, 2004). Query clustering is a task that automatically groups semantically or syntactically related queries in a query repository. Query clustering methods can be broadly divided into two categories: content-based clustering and link-based clustering (*Fu et al.*, 2004). The first category of methods is those that cluster queries using the content therein, which can be represented in different ways, such as characters and keywords. The second category of methods performs the clustering based on the query-document relationship. User sessions are captured by search engines, which record what queries a user submits and which documents a user clicks on afterwards. If two queries are always linked to the same set of documents, they are very likely to be related. If we consider product features as queries, result documents as forum threads, and the occurrence of feature terms in threads as links from queries to documents, then query clustering methods can be intuitively employed in a product feature grouping scenario. Topic models add another latent topic layer to traditional word document models. The idea behind this is that each document can be viewed as a mixture of various topics, where a topic is a probability distribution over words. Hofmann introduced latent topics in his probabilistic latent semantic indexing (pLSI) (*Hofmann*, 1999). Using pLSI, one can assess the probability of generating a word given the topic. However, pLSI is prone to overfitting, and the model learns the topic mixture only for documents in the training data set. Latent Dirichlet allocation (LDA) (*Blei et al.*, 2003) was then proposed by Blei et al. to solve the problems by introducing a Dirichlet prior over topics and words. Griffiths et al. extended the LDA model (*Griffiths and Steyvers*, 2004), and the new version had better results. LDA is adopted as one of the similarity functions in the proposed product feature clustering method.

3.3 Research method

The purpose of this chapter is to identify all the product features, aspects and components that explicitly present in a thread and to group related features into clusters. The whole process follows a series of steps as illustrated in Figure 3.1.

First, discussion threads in the forum database are processed with a part-of-speech (POS) tagging tool. Each word that presents in the data corpus is marked as a particular part of speech, and meanwhile noun phrases are also recognized. MITRE's Alembic Workbench

Table 3.1: Attributes used for summarizing in previous literature

Attribute	**Literature**
Position–At which part of the paragraph does the keyword or topic sentence usually occurs	(*Kupiec et al.*, 1995; *Jiang et al.*, 2009)
Skeleton–Whether the topic sentence is in the title, headline, abstract, introduction or conclusion	(*Kupiec et al.*, 1995; *Jiang et al.*, 2009)
Length–Word count	(*Kupiec et al.*, 1995; *Blair-Goldensohn et al.*, 2008; *Jiang et al.*, 2009)
Co-reference–Linking different name aliases, synonyms and morphological variants within a document	(*Okurowski et al.*, 1999)
Term frequency (TF)–Number of occurrences of the keyword in the document	(*Lin*, 1999; *Hu and Liu*, 2004a; *Kupiec et al.*, 1995; *Okurowski et al.*, 1999; *Jiang et al.*, 2009)
Cue words–Presence of words like "significantly" and "impossible"	(*Kupiec et al.*, 1995)
Uppercase words–Presence of uppercase words	(*Kupiec et al.*, 1995)
Inverse document frequency (IDF)–Logarithm of total number of documents in the collection divided by the number of documents containing the term	(*Okurowski et al.*, 1999; *Lin*, 1999; *Blair-Goldensohn et al.*, 2008)
Numerical data–A boolean value indicating if containing a number	(*Lin*, 1999)
Proper name–A boolean value indicating if containing a proper name	(*Lin*, 1999)
Pronoun or adjective–A boolean value indicating if containing a pronoun	(*Lin*, 1999)
Quotation–A boolean value indicating if containing a quotation	(*Lin*, 1999)
Number of terms–Number of terms contained	(*Conroy and O'leary*, 2001)
Term frequency-inverse document frequency (TFIDF)	(*Jiang et al.*, 2009)
Phrase distribution	(*Jiang et al.*, 2009)
Likeliness–Likeliness of the topic sentence given the document terms	(*Conroy and O'leary*, 2001)
Compactness–How likely is it that terms contained in the product feature appear together	(*Hu and Liu*, 2004a)
Opinion–Whether there is any sentiment-bearing terms in the same sentence or nearby	(*Hu and Liu*, 2004a)
Static patterns–Terms which the feature phrases most likely follow or precede	(*Blair-Goldensohn et al.*, 2008)

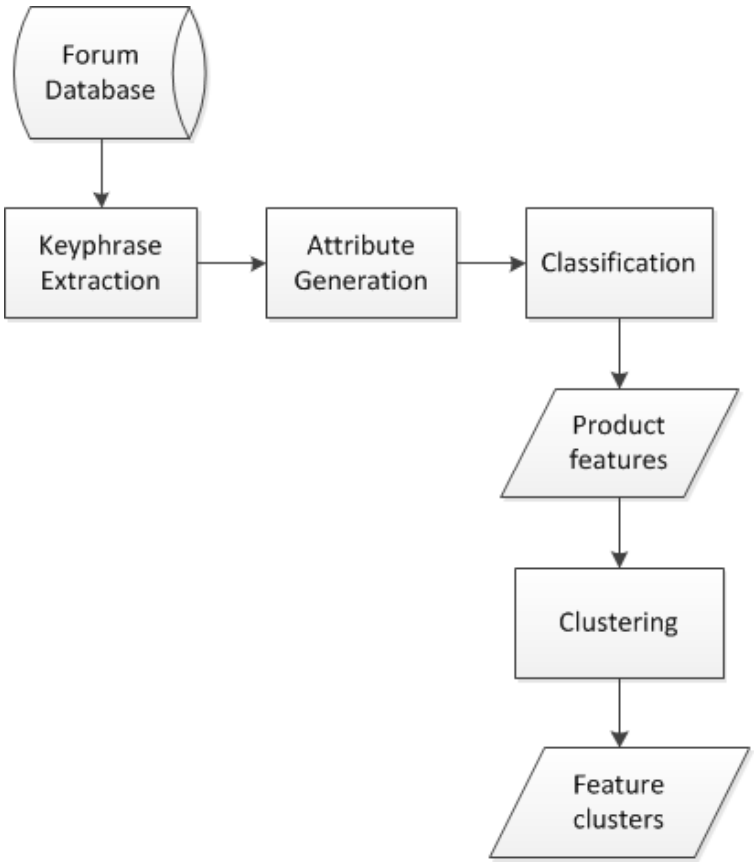

Figure 3.1: Flowchart of product feature extraction and clustering

(*Day et al.*, 1997) is used in the process to annotate terms. After that, only nouns and noun phrases are kept and all the other categories of words including pronouns, verbs, adjectives, adverbs, and prepositions are removed. Next, stop words are eliminated and other terms are stemmed. The extracted noun phrases are then indexed for fast accessing. Next, each candidate product feature is represented as a vector of attributes for characterizing it. A learning algorithm is then employed to help identify the real product features. After product features are extracted at the thread level, these product features are aggregated for forum-level summarization.

The remaining part of this section describes the approaches of product feature extraction and feature clustering separately.

3.3.1 Product feature extraction

Classifiers

Simply speaking, product feature extraction is identifying product features from a set of noun phrases that are extracted from thread content. Therefore, it can be naturally modeled as a binary classification problem. Four alternative classifiers were employed to determine product features: support vector machine (SVM) (*Gunn*, 1998; *Hearst et al.*, 1998; *Chang and Lin*, 2011), neural networks (NN) (*Bishop*, 1995; *Lippmann*, 1987), decision trees (DT) (*Breiman et al.*, 1984; *Quinlan*, 1993), and logistic regression (LR) (*Hosmer and Lemeshow*, 2000).

Attributes

Incorporating some existing attributes from traditional text summarization, the attribute set for product feature extraction consists of 16 attributes. *Term frequency-inverse document frequency (TF-IDF)* is a very important measure of term significance in a document; IDF stands for inverse document frequency and it can be calculated using the following equation:

$$\text{idf}(f) = \log \frac{N - \text{df}(f) + 0.5}{\text{df}(f) + 0.5} \tag{3.1}$$

where N is the total number of threads in the forum and $\text{df}(f)$ denotes the document (thread) frequency of product feature f in a data set. The reason why df is represented in an inverse form is that the more frequently the term appears in different documents, the less meaningful it is. Multiplying by tf gives us the TF-IDF feature:

$$\text{tfidf}(d, f) = \text{tf}(d, f)\text{idf}(f) \tag{3.2}$$

where $\text{tf}(d, f)$ is the term frequency of feature f in thread d.

Where the candidate product feature term first occurs is also an important attribute. It is referred to as *first occurrence position* and has three possible values (1) sub-forum title, (2) thread title and (3) other part in the thread. Apparently, the terms in titles are more important in conveying the user opinions about a product.

Online forums are a special structured data source and each thread is a collection of posts from different forum users. Therefore, there are some special properties we can potentially utilize. The *number of posts* containing the product feature and the *number of distinct users* mentioning the product feature are two of such attributes. If either of the two numbers is large, it is very likely that the discussion is about the product feature.

Average distance between two occurrences denotes the average number of words between two occurrences of a product feature in a thread. It is also an attribute for the product feature term and can help exclude those words or phrases that are too frequently used by a single user or in a single paragraph.

According to (*Hu and Liu*, 2004a,b), emotional adjectives and product features are usually present together in the same sentences to express user opinion. Therefore, *emotional words* are used as a special indicator for product features.

A full list of attributes can be found in Table 3.2.

3.3.2 Product feature clustering

With the evolution of Web searching technologies, the problem of query clustering has attracted substantial attention from researchers and has been extensively studied. The clustering problem concentrates on the queries that Web users enter to launch a search and aims at grouping together different queries used to search for the same desired information. According to the search strategy, most of keywords in a query should be present in search result documents. Considering the product features as queries and result documents as threads, we can use the query clustering techniques to group product features. In the next part of this section, we first introduce similarity functions, which are used to measure the distance between two product features, and then describe a clustering algorithm to group close features together.

Similarity functions

The performance of a clustering algorithm largely depends on the selection of distance function. Four alternative similarity functions were developed based on different types of information, and the distance between two product feature terms is calculated as one minus the similarity. The first two similarity functions are defined based on the syntactic constituent of the product feature terms but at different levels. The third one is called co-occurrence similarity based on the feature term containment relationship, which can reveal the semantic

Table 3.2: Attributes of product features

Attribute	**Symbol**	**Description**
TF	A_1	Frequency of the phrase in the thread
	A_2	A_1 divided by the total number words in the thread
DF	A_3	Document frequency of the phrase in the collection
Compactness	A_4	For a phrase containing more than one word, how likely is it that the words to present together
Emotional Words	A_5	Whether there is an emotional word close by
TF-IDF	A_6	Term frequency-inverse document frequency of the phrase
Number of Posts	A_7	Number of posts containing the phrase
	A_8	A_7 divided by total number of posts in the thread
Number of Users	A_9	Number of distinct users mentioning the phrase
	A_{10}	A_9 divided by total number of users participating the thread discussion
Word ratio	A_{11}	Number of words in the phrase multiplying TF divided by total number of words in the thread
Thread Length	A_{12}	Total number of words in the thread
Number of Words	A_{13}	Number of words in the phrase
First Phrase Position	A_{14}	First occurrence of the phrase divided by the total number of words in the thread
First Word Position	A_{15}	First occurrence of the earliest occurring single word in the phrase divided by the total number of words in the thread
Top Word Frequency	A_{16}	Frequency of the most frequent single word in the phrase
User Post Count	A_{17}	Post count of the first user who mentions the phrase
User Thread Count	A_{18}	Thread count of the first user who mentions the phrase
Skeleton	A_{19}	Whether the phrase is in forum name
	A_{20}	Whether the phrase is in thread title
Part-of-Speech	A_{21}	Contains a functional word e.g., "there"?
	A_{22}	How many words are nouns in the phrase
	A_{23}	Contains an adjective?
	A_{24}	Contains a verb?

relationship between terms. The last one is a topic similarity function based on LDA topic distribution.

1. Character-based similarity

The character-based similarity function (*Wen and Zhang*, 2002) is defined as

$$\text{sim}_{\text{char}}(f_1, f_2) = 1 - \frac{\text{edit-distance}(f_1, f_2)}{\max(\text{len}(f_1), \text{len}(f_2))} \tag{3.3}$$

where $\text{len}(f)$ is the character length of feature term f and $\text{edit-distance}(f_1, f_2)$ is the edit distance (*Gusfield*, 1997) between two feature terms. If the cost of converting one term to another is low, it is very likely they share the same root word and have the same or very similar meaning. The following three product features, which are normally used to refer to the same product aspect, can be grouped using this function: "auto tran", "auto tranni", and "auto transmiss".

2. keyword-based similarity

The keyword-based similarity function (*Wen and Zhang*, 2002; *Fu et al.*, 2004) is defined as

$$\text{sim}_{\text{keyword}}(f_1, f_2) = \frac{\sum_{i=1}^{k} \left(\text{cw}_i(f_1) \times \text{cw}_i(f_2)\right)}{\sqrt{\sum_{i=1}^{m} w_i^2(f_1)} \sqrt{\sum_{i=1}^{n} w_i^2(f_2)}} \tag{3.4}$$

where cw_i is the weight for the ith common term shared by f_1 and f_2, and $w_i(f)$ is the weight for the ith term in f. The TF-IDF method is adopted to weight terms in f. If the most important part of two terms is the same, they should be grouped together. This function is similar to the character-based similarity function but at the word level. An example is "Honda transmission" and "transmission", where Honda appears in almost all the threads and can be ignored.

3. Co-occurrence similarity

The co-occurrence similarity function (*Wen and Zhang*, 2002; *Fu et al.*, 2004) is defined as

$$\text{sim}_{\text{co-occur}}(f_1, f_2) = \frac{\text{cdf}(f_1, f_2)}{\max\left(\text{df}(f_1), \text{df}(f_2)\right)} \tag{3.5}$$

where $\text{cdf}(f_1, f_2)$ is the common document frequency of f_1 and f_2, and $\text{df}(f)$ is the document frequency of f. Figure 3.2 illustrates product feature co-occurrence.

In the example, airbag and SRS light are two Honda features that usually present together in online forums. Actually, SRS is short for supplemental restraint system and is vitally important for drivers. If the SRS light is on, it probably means the airbags or the seat belts are worn out. Therefore, we can see that the two features are both related to the airbags and safety system, which should be grouped together. The co-occurrence similarity function can reveal this kind of information for us.

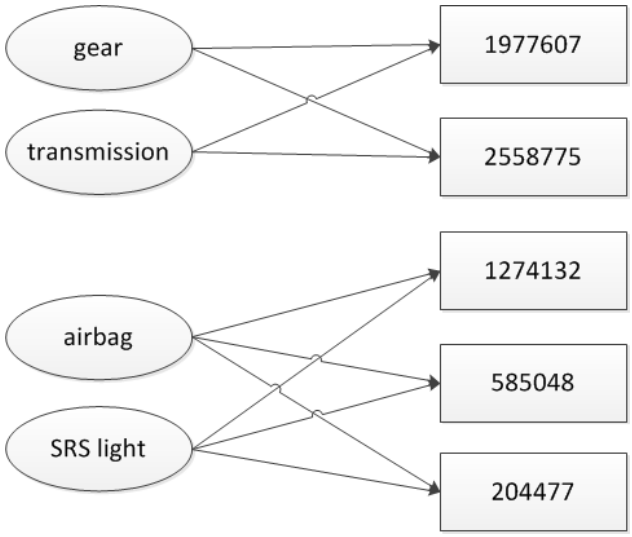

Figure 3.2: Product feature co-occurrence example

4. LDA similarity

Latent Dirichlet allocation (LDA) (*Blei et al.*, 2003) was a generative probabilistic model presented by Blei et al. It posits that each document in a text collection is modeled as a finite mixture over a set of documents and each topic is modeled as an infinite mixture over a set of topic distributions. To gain a better result, Griffiths et al. extended Blei's work with the addition of a Dirichlet prior (*Griffiths and Steyvers*, 2004), which was called smoothed LDA and was mostly used later on.

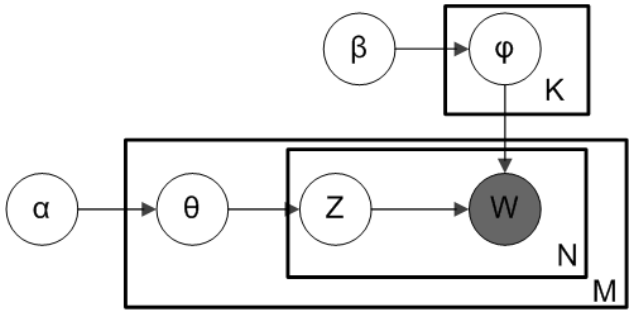

Figure 3.3: Plate notation for smoothed LDA

The algorithm of smoothed LDA that we chose is employed as an alternative method for product feature similarity measuring. The plate notion for smoothed LDA is shown in Figure 3.3. To be consistent with the original paper, we use the same notions here. Let α be the parameter of the uniform Dirichlet prior on the per-document topic distribution, β be the parameter of the uniform Dirichlet prior on the per-topic word distribution, θ_j be the topic distribution for document j, φ_i be the word distribution for topic i, $Z_{j,t}$ be the topic for the

tth word in document j, and $W_{j,t}$ be the specific word. Thus, the total probability of the model is

$$P(W, Z, \theta, \varphi|\alpha, \beta) = \prod_{i=1}^{K} P(\varphi_i|\beta) \prod_{j=1}^{M} P(\theta_j|\alpha) \prod_{t=1}^{N} P(Z_{j,t}|\theta_j) P(W_{j,t}|\varphi_{Z_{j,t}})$$

In their paper (*Griffiths and Steyvers*, 2004), the authors also gave an inference technique called Gibbs sampling to learn the various distributions and estimate the parameters in the above model, which is also applied in our task. Based on the term-topic distribution, we propose the following LDA similarity function:

$$\mathrm{sim}_{\mathrm{lda}}(f_1, f_2) = \frac{\sum_{i=1}^{k} \left(\mathrm{w}_i(f_1) \times \mathrm{w}_i(f_2)\right)}{\sqrt{\sum_{i=1}^{k} w_i^2(f_1)} \sqrt{\sum_{i=1}^{k} w_i^2(f_2)}} \tag{3.6}$$

where k is number of topics and w_i is the term-topic distribution.

Wordnet (co-occurrence in synset) (*Miller et al.*, 1995) similarity may also be used for measuring term similarity. Please note that Wordnet was not used in the current study as it may introduce false positives if terms are not correctly disambiguated (we did not use word-sense disambiguation).

So far, four similarity functions have been described. They all have their own advantages at finding a certain type of related feature terms. To integrate these advantages, the above similarity functions can be combined with a linear combination as follows:

$$sim = \alpha \cdot \mathrm{sim}_{\mathrm{char}}(f_1, f_2) + \beta \cdot \mathrm{sim}_{\mathrm{keyword}}(f_1, f_2) + \gamma \cdot \mathrm{sim}_{\text{co-occur}}(f_1, f_2) + \delta \cdot \mathrm{sim}_{\mathrm{lda}}(f_1, f_2) \tag{3.7}$$

For simplicity, in this study, we used equal weight to combine the four similarity scores.

Cluster algorithm

With similarity functions, the similarity of each pair of extracted feature terms can be calculated and their distance is represented as one minus similarity. Next, a clustering algorithm is needed for executing the clustering. DBSCAN (*Ester et al.*, 1996) is a density-based clustering algorithm and experiments showed that it was an effective algorithm that outperformed most other methods (*Wen and Zhang*, 2002). Thus, we opted to use DBSCAN as the feature clustering algorithm.

3.4 Experimental results

This section presents the evaluation questions that guided the design of our experiments. The data sets and how they were collected are also described. After that, the metrics for

measuring the system performance are given. Finally, the experiment details and results are discussed.

3.4.1 Evaluation questions

The questions we used to evaluate the proposed method are as follows:

Question 7. *How effective are the attributes in measuring the significance of a product feature term in a thread?*

Question 8. *What is the performance of the learning algorithm in product feature extraction?*

Question 9. *How does each similarity function perform?*

Question 10. *Is the linear combination of similarity functions better than individual similarity function?*

Question 11. *How well does the product feature clustering work?*

We then sought to answer these questions through a series of well-designed experiments.

3.4.2 Data set

For the sake of simplicity, a sub-forum of Honda-tech with the forum name Honda Accord (1990-2002), was selected as the experiment data set; it contains 65,084 threads, 476,738 posts and 32,238 distinct users that participated in the discussion.

To evaluate the algorithm of product feature extraction, a set of 300 threads was sampled from the Honda Accord sub-forum with a random selection of up to 20 noun phrases from every thread. The same group of domain experts (see Section 2.4.2) were employed to select the product features or topic key phrases from the 20 candidates. Based on the input of the human judges, 1,102 product features were collected with an equivalent number of non-feature phrases as negative examples.

The evaluation of product feature clustering is nontrivial. A total of 197,249 unique noun phrases were extracted from 32,238 threads. Considering the large amount of product features, it is almost impossible to go through all of them and cluster them into feature groups with high accuracy. A sample set of 100 groups of product features was produced by each clustering algorithm and then human judgments were solicited based on the following question: whether or not a product feature belongs to the cluster.

3.4.3 Evaluation metrics

The metrics of precision, recall, and F_1 that have been used in product defect detection also can be adopted here to measure the performance of product feature extraction, but the definitions of these metrics need to be changed a little to adapt to the new context:

- Precision is defined as the percentage of the correctly extracted product features over the total number of noun phrases extracted.
- Recall is defined as the percentage of correctly extracted product features over the total number of product features, regardless of whether they were extracted or not.
- F-measure is still defined as the weighted average of precision and recall.

For product feature aggregation, we opted to use the same metrics that Wen et al. (*Wen and Zhang*, 2002) used in solving the query clustering problem, which are described below:

- Precision–the ratio of the number of correctly grouped product features to total number of product features in a feature group
- Normalized recall–the ratio of the number of correctly grouped features to the maximum number of correctly grouped features in the sample feature set.

The reason we used a normalized recall is that there is no standard feature group available and an alternative option has to be applied.

3.4.4 Product feature extraction

As the baseline method, we chose to use the same rule-based method as was used in (*Hu and Liu*, 2004a), which also aimed to extract product-related key phrases and used a very similar data source: customer reviews. However, extracting product features from product forums is more challenging than doing so from customer reviews. For one, when writing a review, most customers want to describe the advantages or disadvantages of a product, and that is usually their only intention; on the other hand, customers who post in product forums might have various intentions. Sometimes they review a product, as would typically happen in a customer review, but more frequently they may ask questions about product usage or seek help with troubleshooting problems. Topic drift and terminology change make it challenging to extract product features from online forums. Moreover, reviews are usually short, concise and focused on product features. Therefore, patterns can be easily detected and naturally applied to discover more feature terms. A forum thread, however, usually starts from a story, and users try their best to provide more detailed information in order to get help from others. This tendency toward storytelling can introduce noise into the feature

extraction process. In the baseline method, product features are extracted step by step and a list of techniques/attributes are applied:

- DF
- compactness/p-support
- nearby opinion word identifier
- sentiment

A set of ad-hoc thresholds were manually selected at each step and no supervised learning method was used. Since the thresholds cannot be directly applied in this scenario, for this experiment, we used a decision tree classifier as the classification method to help with threshold selection. Beyond that, we also compared the approach with a topic modeling method, LDA (*Blei et al.*, 2003), and a supervised learning method, KEA (*Witten et al.*, 1999) to get a comprehensive understanding of the performance of our product feature extraction system.

Table 3.3 shows the classification results of four alternative classifiers with enriched feature sets.

Table 3.3: Product feature extraction results

	Accuracy	Precision	Recall	F_1
LDA	0.591	0.646	0.400	0.494
KEA	0.344	0.375	0.468	0.417
Hu	0.637	0.646	0.628	0.641
DT	0.657	0.667	0.632	0.649
SVM	0.691	0.705	0.658	0.681
LR	**0.7**	**0.707**	**0.681**	**0.694**
NN	0.669	0.651	0.731	0.689

According to the results, the logistic regression method produced the best results among all the methods, with accuracy equal to 0.7 and F_1 equal to 0.694, thereby outperforming all the baseline methods. Even compared to the best method of the baselines, there was still a 8.9% gain. Table 3.4 lists the top 10 attributes with the highest weights in the fitted logistic regression model, where positive coefficient values indicate that the attributes have positive effects on the odd ratio of a candidate phrase being a product feature, while negative values indicate the effects are negative.

The first two attributes in the list have high weights but opposite signs. Since word ratio has relatively higher weight, it will dominate the value: if the phrase contains multiple words and has relatively high frequency in the thread, it is highly likely to be a product feature. Other salient attributes include part-of-speech attributes and skeleton attributes: the former means number of non-functional words are good indicators of product features; the latter means most product features may be found in the thread titles.

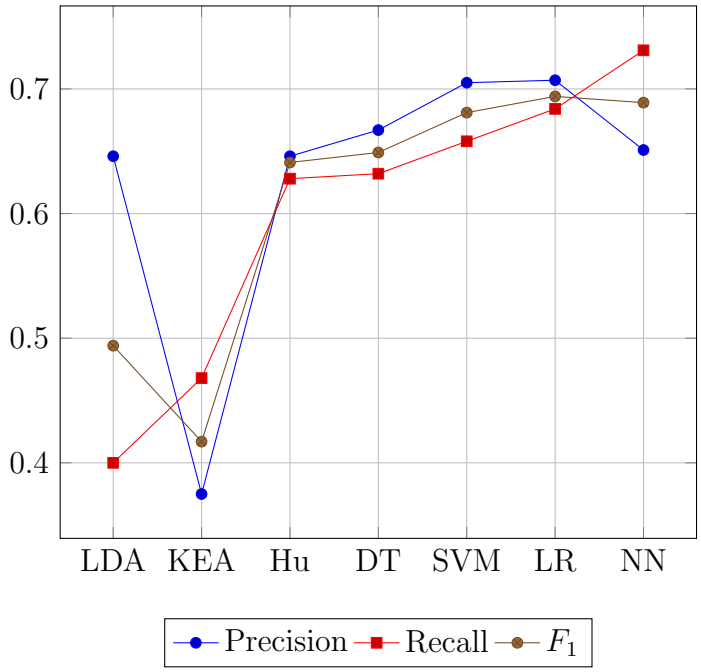

Figure 3.4: Product feature extraction results

Table 3.4: Top 10 attributes in logistic regression model

Attribute	Coefficient
Word ratio (A_{11})	87.5727
TF (A_2)	-86.0164
Number of Posts (A_8)	2.3415
Part-of-Speech (A_{21})	-2.1461
Number of Words (A_{13})	-1.5521
Skeleton (A_{19})	-0.8284
First Word Position (A_{15})	-0.8098
Part-of-Speech (A_{22})	0.739
Skeleton (A_{20})	0.5121
Compactness (A_4)	-0.4827

3.4.5 Product feature clustering

DBSCAN is a parametric clustering model, which requires two parameters: minimum number of points per cluster (MinPts) and ϵ. A straightforward parameter, MinPts controls the number of clusters produced; ϵ is the maximum distance of two points being grouped together. The DBSCAN algorithm starts from an arbitrary one-point cluster and the points within the ϵ-neighborhood are capable of being added to it. Therefore, higher ϵ generates higher recall but lower precision. In our experiments, MinPts was always set to 2, which means individual features cannot become clusters, whereas the ϵ of each cluster method was manually determined based on the cluster results of a sample of 1,000 key phrases.

1. Clustering using a character-based similarity function

   In online communities, it is normal for users to generate content with spelling errors, and there is no proofreading system to check or correct their spelling. Character similarity function is effective in detecting different variations of the same term. Figure 3.5 shows four of the result clusters. In this experiment, ϵ was given as 0.16.

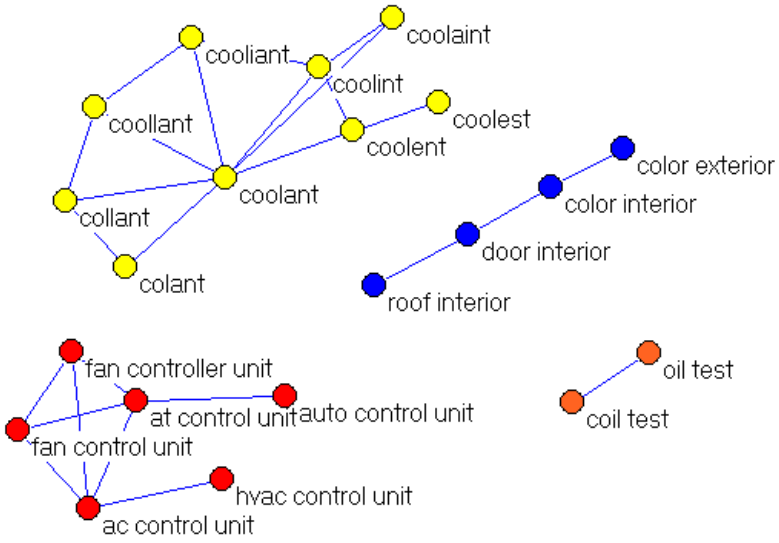

Figure 3.5: Character similarity clustering results

In the top left cluster, “colant”, “collant”, “coolaint”, “coolent”, “cooliant”, “coolint” and “coollant” are misspelled variations of “coolant” that are correctly captured by edit-distance and mapped to the correct spelling. However, “coolest” is a mis-clustering. Similarly, in the bottom right cluster, although “oil test” and “coil test” look very similar in shape with only one character different between them, they are two different terms and should not be grouped together. This is the main cause of false positives in character similarity clustering.

2. Keyword using a character similarity function

   When two phrases share the same terms and the different part is trivial, it is likely that the two phrases express the same underlying concept. Figure 3.6 illustrates some examples of clustering results using a keyword similarity function, which ϵ was set to 0.2.

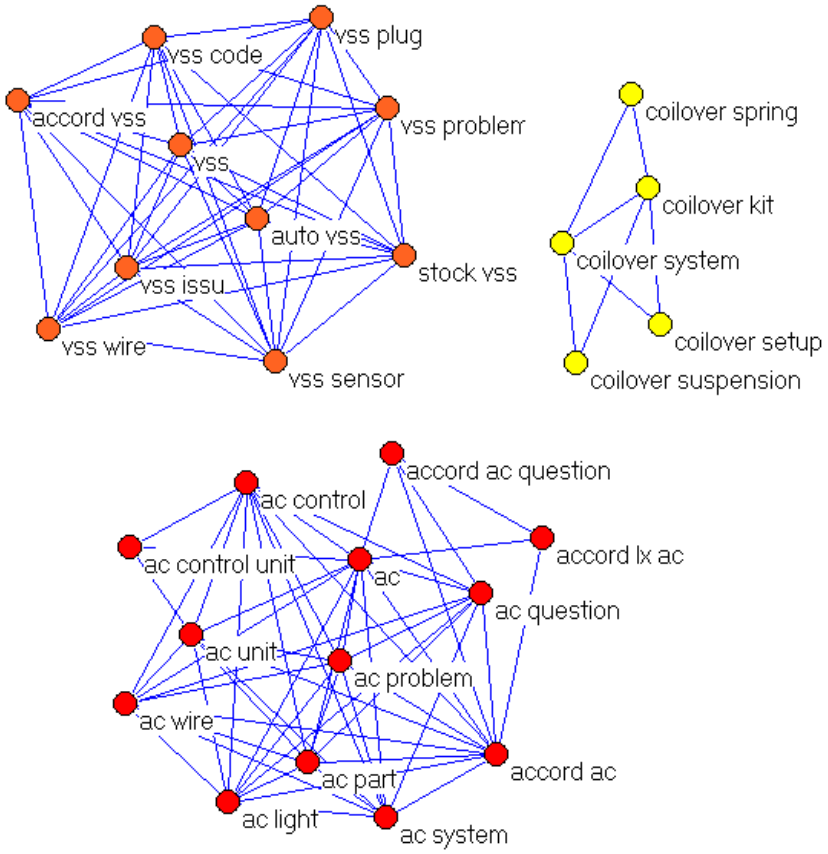

Figure 3.6: Keyword similarity clustering results

In the top left cluster, "ac" is a much more important term compared to "light", "system", "wire", "part", etc., and it is a core product feature that connects a set of other features. If we look at the features that are directly connected, such as "vss" vs. "vss code" and "vss" vs. "vss wire", but "vss code" and "vss wire" seem to be a little bit far away. Topic drift is a problem that may cause two not quite related terms to be grouped together.

3. Clustering using a link-based similarity function

   Link similarity function can be used to find terms that are not similar in shape but have underlying hidden connections. Figure 3.7 shows some clustering results based on co-occurrence with ϵ equal to 0.2.

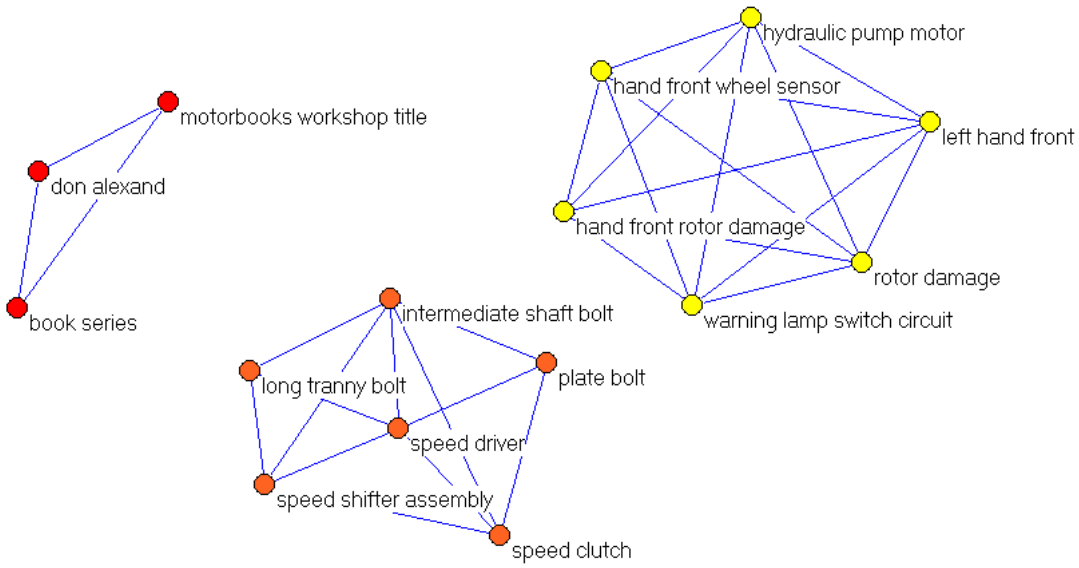

Figure 3.7: Link similarity clustering results

In the first cluster, the three keywords are all extracted from the same piece of information about a handbook called "High-Performance Handling Handbook" that one forum contributor shared, and part of the book description was quoted multiple times. It is hard to say that these keywords are not related, but quotations are not the information sought and is one of the limitations of linked-based similarity. The example cluster at the bottom is related to speed and transmission, where the components are connected within the thread scenarios. However, when put in a wider context, the connection among them is weak.

4. Clustering using LDA-based similarity function

   As the first step of LDA analysis, the number of topics must be determined. The perplexity on a hold-out data set was calculated; when the number of topics was 25, perplexity reached the minimum (Figure 3.8). Therefore, in the LDA model, the number of topics was set to 25.

   What follows is part of the clustering results from the LDA-based similarity function.

   In Figure 3.9, the first example is related to muffler, intake, and exhaust system. Although it covers many different car parts and contains lots of different terms, we still can clearly see that all the features and components are highly correlated without looking into the thread context; therefore, LDA is a better similarity function compared to the others, at least with respect to the data set presented here.

5. Clustering using combined similarity function

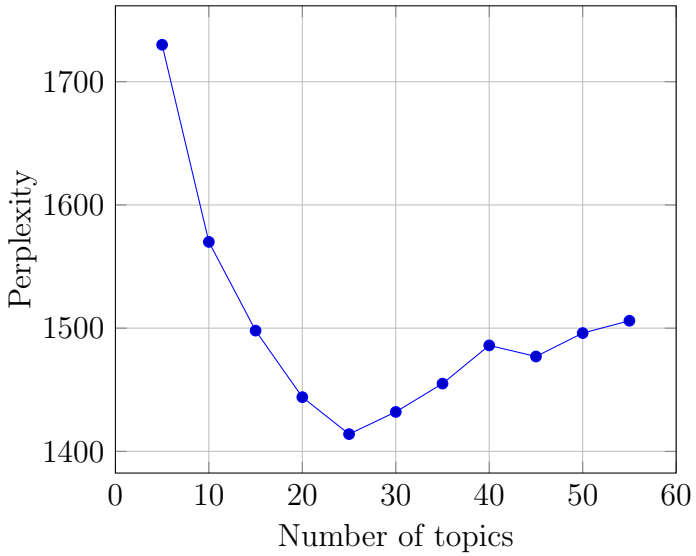

Figure 3.8: LDA perplexity on feature terms

As a combined similarity function, it incorporates all the advantages of the above four similarity functions and should produce much better results.

Some clustering results using combined similarity function are shown in Figure 3.10, where ϵ was set to 0.35 and even weights (0.25) were given in Equation 3.7. As we can see from the results, it also generates a cluster of "coilover", but more related terms were detected as compared to the results of the keyword similarity function. Since the threshold on keyword similarity alone was even lower ($0.35 * 0.25 = 0.0875 < 0.2$), most of the additional terms must be due to a combined effect of keyword similarity with other similarities. From these examples presented, we can see that using a combined similarity function is much better than using any one similarity function alone.

Table 3.5 shows a summary of the clustering results of DBSCAN combined with different similarity functions.

Table 3.5: Product feature clustering results

Similarity Function	Precision	Recall	F_1
Character	0.4067	0.9160	0.5633
Keyword	0.4615	0.9500	0.6213
Link	0.4216	0.9998	0.5931
LDA	0.6584	0.9746	0.7859
Combined	**0.8624**	**0.9261**	**0.8932**

According to these results, recall seems to be extremely high for all the methods; this is mostly due to how recall is calculated. It is almost impossible to find all the instances

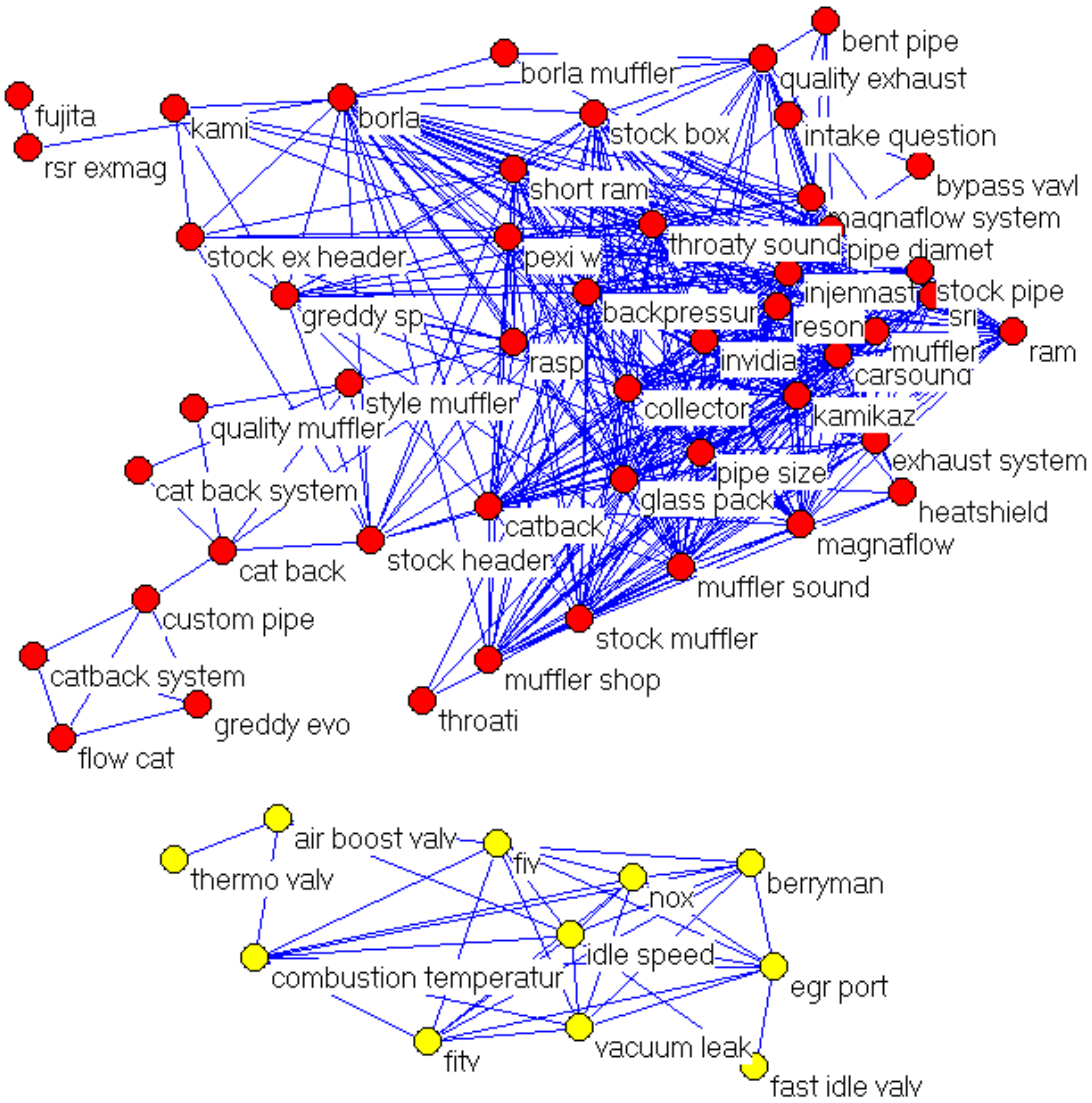

Figure 3.9: LDA similarity clustering results

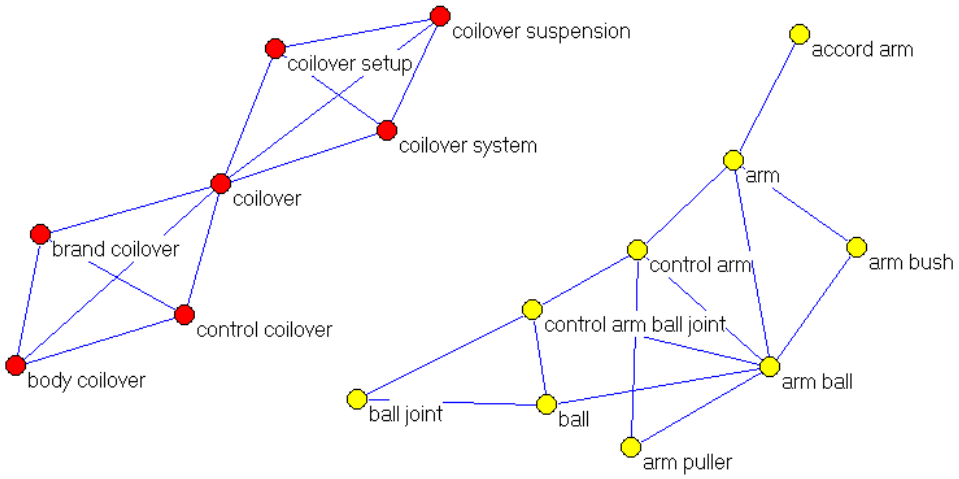

Figure 3.10: Combined similarity clustering results

of each cluster, and the few false- negative examples are mostly from the results of other similarity functions. The link-based similarity method produced close to 1 recall, and this is because it can find some clusters that other functions could not detect, which does not necessarily mean it is better than the other methods. LDA in general outperformed all the other individual similarity methods in terms of F_1, and the feature terms grouped by LDA are highly relevant. Lastly, combined similarity function is the best, with a precision equal to 0.86, which is consistent with our observations, and its performance is strong.

3.5 Conclusion

The solutions to two problems were proposed in this chapter: product feature extraction and product feature clustering. Product feature extraction is a challenging problem and was initially undertaken in the situation of online reviews. In this chapter, it was converted to a classification problem and a richer feature set was collected. The performance of our proposed method is acceptable and better than baseline methods with around a 9% gain in F_1, but there is still room to improve. On the other hand, a clustering method was proposed to solve the product feature clustering problem with five different similarity functions. The model built based on combined similarity function produced the best results, with up to 0.86 precision and 0.92 recall, which is promising.

Looking forward, there is still room to improve product feature extraction. The quality of candidate phrase generation can be improved by extraction process refinement and algorithm tuning. One experiment of testing the robustness and utility of the method we have proposed is to apply it to online forums in another industry, e.g., iPhone forum. Such cross-industry

application of the model would show how the model performs and what features remain effective across industries.

Chapter 4

Learning to Rank Threads

One of the fundamental challenges in online forums is thread retrieval. With the growth of social media, online forums are accumulating vast volumes of informational exchanges from community members. How to quickly and precisely find useful information from millions of discussion posts has become a critical problem. In the past decade, a lot of effort has been made toward improving search capability and thread retrieval; however, few proposed methods of retrieval can provide high-quality search results to meet users' informational needs. In this chapter, we adapt learning to rank technologies and propose a novel ranking algorithm for retrieving highly authoritative and related threads in online forums.

4.1 Introduction

As online forums multiply and attract more participants, there is an increasing demand for efficient forum archive searching. Different from traditional document ranking, thread ranking has its own characteristics. Usually, each forum search is conducted within a specific online forum, and user queries are mostly from the same domain. As a result, user information needs can be very specific, and content meeting those needs may be sparse and difficult to discover. Moreover, each thread is a collection of posts from different users, whereas most traditional documents are drafted by a single person. Therefore, thread content may not be very coherent and uniform, and post quality is largely determined by contributing authors' knowledge and expertise.

To address the thread ranking problem, we must first figure out what the characteristics of a good thread are. Most of us have experienced disappointing results in online searches. When we have a question and query it in an online forum, the top results produced by the search engine may not contain much information that is useful or authoritative. Figure 4.1 lists the results of a query "How to put a CheckBox in a DataGrid?" in a vbCity forum search.

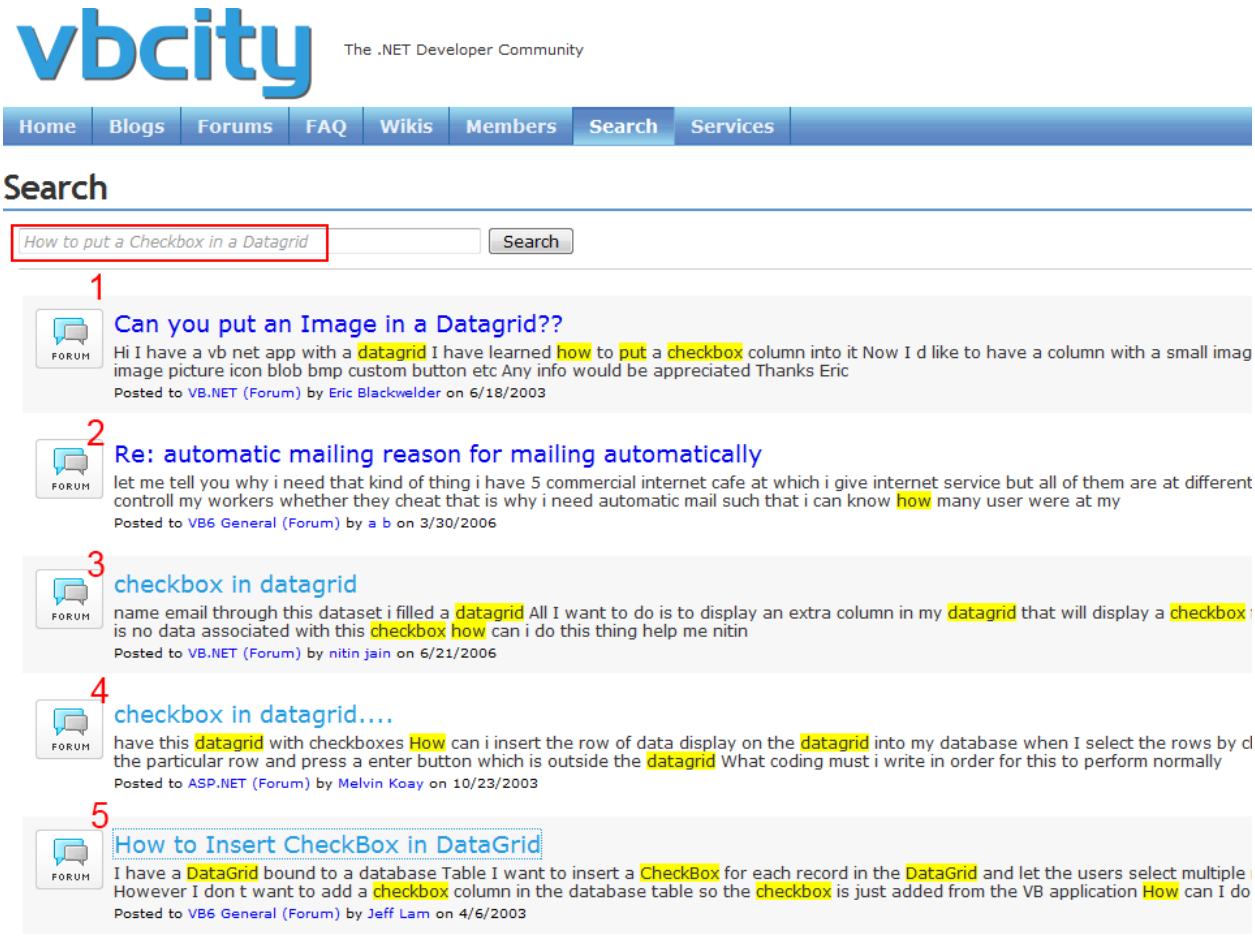

Figure 4.1: Search results from vbCity for query "How to put a CheckBox in a DataGrid?"

As we can tell from the titles and snippets, the first two threads are not related to the query, but they are ranked at the top. The third and fourth threads are quite relevant, and the fifth one is perfect. If we click on the links and look into the thread content, we will see that the third thread contains some useful information but the fourth one is a thread with no reply, which is not helpful at all. In the fifth thread, the problem is resolved. So the ideal order of these five threads should be $5 > 3 > 4 > 1 > 2$. This example is just one of the queries in our case studies. For most of the queries, none of the top 5 results is a good match. There are two main reasons for this: (1) the result threads do not match or are not directly related to a user's query; (2) even though a search engine search may return some related threads, there is no resolution or sufficient useful information contained in them. This second point tells us that relevance is not the only factor we need to consider when building a successful search engine for online forums. According to the guidelines for result page rating (*Schwartz*, 2008), we can see that Google had similar considerations when designing their Internet search engine. The variables of relevance and usefulness can be considered together as authoritative scale levels (ASLs), defined in Table 4.1.

Table 4.1: Authoritative scale levels

ASL	**Definition**
3	This denotes discussion threads that are perfectly related to the query topic, neither too broad nor too specific, and contain high-quality and authoritative information for resolving the problem.
2	This includes two categories of discussion threads: 1) the threads are perfectly related to the query topic but contain very little useful or authoritative information; 2) the threads contain valuable and authoritative information but are only loosely related to the query topic.
1	This denotes discussion threads that are related to the query topic but have no authoritative or valuable information that might be leveraged to resolve the problem.
0	This denotes discussion threads that have no relevance or are only marginally related to the query topic.

Given this definition of ASLs, we state the research question central to this chapter as follows: Given the large volume of data generated by online forum participants, how can we design a better search method that allows users to locate highly relevant and authoritative knowledge?

4.2 Literature review

The literature review of this chapter contains two parts. The first part focuses on research into learning to rank, and the second part discusses previous studies on thread ranking.

4.2.1 Learning to rank for information retrieval

Learning to rank is a type of machine learning problem, in which the goal is to construct a ranking model for information retrieval and other purposes. According to how the ranking problem is formulated, existing algorithms can be grouped into three categories: pointwise approaches, pairwise approaches and listwise approaches. Some representative algorithms of each category are listed in Table 4.2.

Table 4.2: A partial list of learning to rank algorithms

Category	Algorithms
Pointwise	PRank (*Crammer and Singe*, 2001)
	McRank (*Li et al.*, 2007)
Pairwise	RankSVM (*Herbrich et al.*, 1999)
	RankBoost (*Freund et al.*, 2003)
	RankNet (*Burges et al.*, 2005)
Listwise	RankGP (*Yeh et al.*, 2007)
	NDCG Boost (*Valizadegan et al.*, 2010)

The three categories of approaches are mainly different in the following four areas (*Liu*, 2009): input space, output space, hypothesis space, and loss function. More details about these algorithms are discussed in the rest of this section.

Pointwise

Learning to rank constructs a model for predicting a relevance degree for each object, and then all objects are sorted based on these degrees. In pointwise approaches, the exact relevance degree is the target to predict. The *input space* consists of each individual document, which is typically represented as a feature vector ($\vec{x}$), and the *output* is an associated ordinal score or a numeric value (y) as the ground truth label. These labels are collected separately by human judges or automatically through some sort of mapping, e.g., movie review rating. A scoring function is defined as a predictor of the relevance degree ($\hat{y} = f(\vec{x})$), which is used as the *hypothesis*. Machine learning techniques, such as multi-class classification, regression, and ordinal regression, are applied. The *loss function* $L(y, \hat{y})$ is also defined at each individual document level.

PRank is an algorithm based on ordinal regression (*Crammer and Singe*, 2001), in which a set of k thresholds are defined ($b_1 \leq \ldots \leq b_{k-1} \leq b_k = \infty$), and documents are naturally projected into the categories delimited by these bounds using a scoring function $f(\vec{x}) = \min_{r \in \{1,\ldots,k\}}\{r : \vec{w} \cdot \vec{x} - b_r < 0\}$. The rank loss is measured as $\sum_{t=1}^{T} |\hat{y}_t - y_t|$, where T is the size of the testing set.

The problems of pointwise approaches are obvious. First, all the algorithms in this cate-

gory process each document individually, and the relative order between documents is never considered, although the nature of the problem itself is ranking. Second, a fact is ignored that documents are associated to certain queries and they are not independent. Last but not least, the loss function can be easily dominated by queries with relatively large numbers of relevant documents in the training data set when the number of relevant documents per query is different.

Pairwise

Pairwise approaches do not care about each individual document. Instead, they focus on predicting the relative order of each pair of documents and reduce the ranking problem to a ranking order classification problem. Compared to pointwise approaches, they are one step further towards the essence of ranking. The *input space* consists of document pairs, and the *output* is a binary value indicating whether one document is more preferred than the other. The *hypothesis* is also different from pointwise approaches and is defined as a preference function $\hat{y}_{i,j} = f(\vec{x}_i, \vec{x}_j)$. Correspondingly, the *loss function* is defined to measure the preference prediction results as $L(y_{i,j}, \hat{y}_{i,j})$.

An example of the pairwise approach is RankNet (*Burges et al.*, 2005), which was reported to have been used in some commercial search engines. In RankNet, suppose $\vec{x}_i$ and $\vec{x}_j$ are two documents associated with the same query and P_{ij} is the posterior of $\vec{x}_i$ ranking on top of $\vec{x}_j$. P_{ij} is modeled as

$$P_{ij} = \frac{\exp(f(\vec{x}_i) - f(\vec{x}_j))}{1 + \exp(f(\vec{x}_i) - f(\vec{x}_j))}$$

The cross-entropy loss is defined as

$$L(f; \vec{x}_i, \vec{x}_j, y_{ij}) = -\bar{P}_{ij} \log P_{ij} - (1 - \bar{P}_{ij}) \log(1 - P_{ij})$$

where $\bar{P}$ is the target probability from the ground truth labels ($\bar{P}_{ij} = 1$ denotes $\vec{x}_i$ ranks higher than $\vec{x}_j$ and $\bar{P}_{ij} = 0$ denotes the opposite). RankNet adopts a network model to solve the problem and uses gradient descent as the optimization method. Experimental results of RankNet showed improvement over PRank, the pointwise approach.

Although pairwise approaches emphasize the relative order between documents, which is ignored in pointwise approaches, the other problem–that queries with a larger number of relevant documents dominate–is still there. In addition, considering the fact that the number of document pairs can be quadratic to the number of documents, the problem of big query domination becomes even more severe and the computation cost also dramatically increases.

Listwise

It is quite straightforward to come out with the idea of directly using the rank itself as the optimization target; however, it is not as trivial as might be expected. Information retrieval

measures such as NDCG (normalized discounted cumulative gain) and MAP (mean average precision) can be potentially used as target functions, but they are not continuous or differentiable (*Xu et al.*, 2008). However, continuity and differentiability are two properties that are required by most existing optimization techniques. To tackle this problem, some studies have sought to approximate the information measures to their continuable and differentiable variant, while other attempts sought techniques that do not have this limitation.

RankGP (*Yeh et al.*, 2007) is one of the recent algorithms that adopt genetic programming to optimize the rank. In RankGP, the ranking function is represented as a tree, where leaf nodes correspond to feature values or constants and non-leaf nodes correspond to mathematical operators, e.g., summation and multiplication. Genetic operators such as mutation, crossover, and reproduction are applied to evolve ranking functions to a new generation. Evaluation results showed that RankGP was competitive with RankSVM and RankBoost, the pairwise approaches.

4.2.2 Online forum thread retrieval

Along with the growth of online forums has come a lot of research dedicated to thread retrieval. An early study on this topic was (*Xi et al.*, 2004), in which the authors proposed a feature-based post ranking function and perceived significant improvement over Okapi BM25 (*Robertson*, 1997). Compared to threads, posts are usually short and lack of context. They may not carry sufficient useful information. A recent study (*Elsas and Carbonell*, 2009) sought to use the structure information and constructed a set of ranking functions based on a language model. The study demonstrated that using the content of a selection of posts is better than the entire thread in terms of ranking. Another work (*Seo et al.*, 2009) tried to utilize thread structure to improve search results.

Our work is different from previous research in three aspects: (1) instead of looking only at document relevance, we also take authoritativeness into consideration; (2) we introduce argument quality and source credibility features into ranking; (3) we apply listwise learning to rank to thread retrieval.

4.3 Research method

This section presents our proposed authoritative ranking method in detail. First, the system architecture is presented to give an overview of the method. Second, two types of features used in the ranking function are discussed: argument quality features and source credibility features. Next, machine learning techniques GA (genetic algorithms) and GP (genetic programming) are described to optimize the ranking function and the search for an optimal solution. Lastly, new fitness functions are presented with the goal of improving the efficiency and performance of GA and GP.

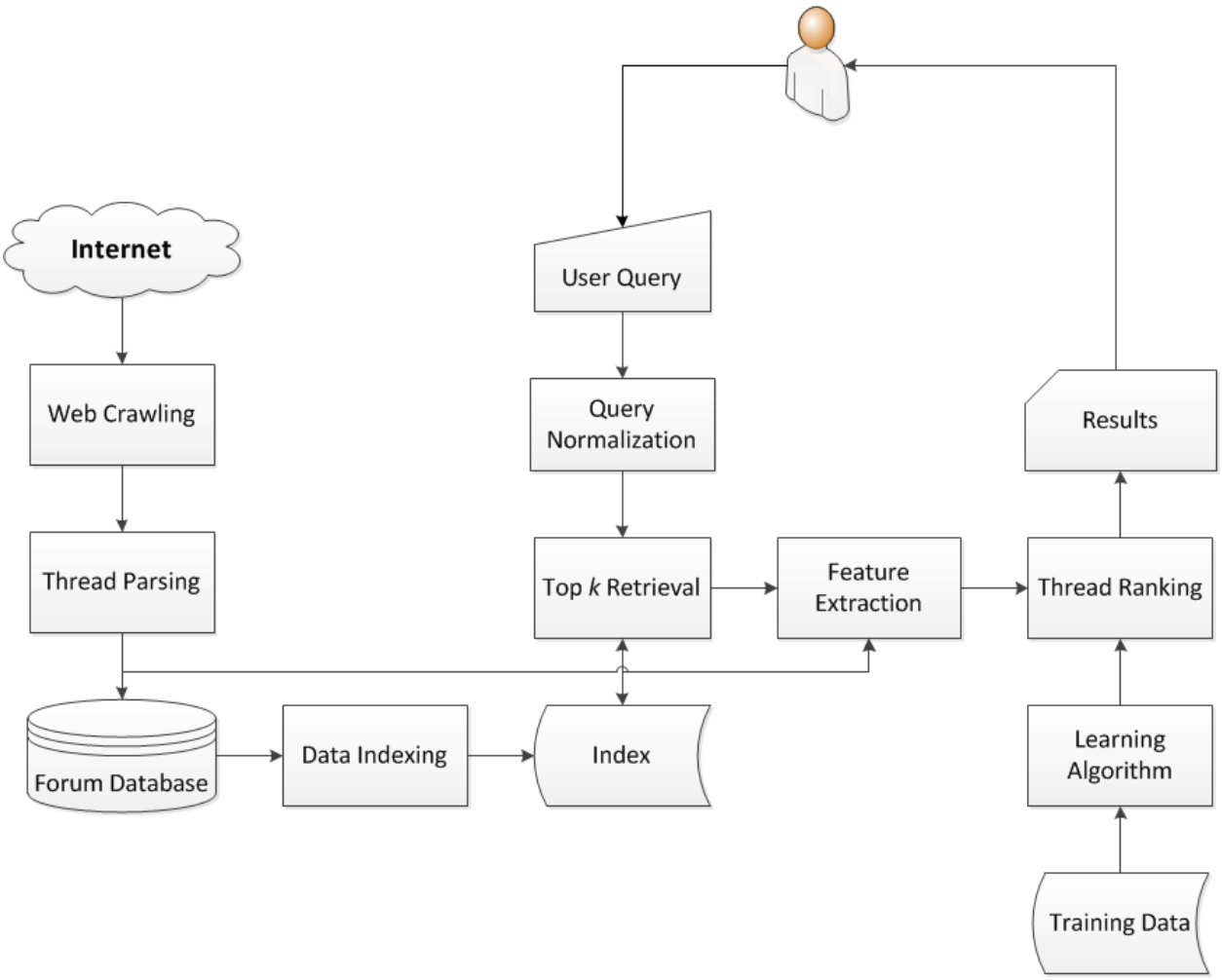

Figure 4.2: Authoritative ranking system architecture

4.3.1 System architecture

The system architecture of authoritative ranking is shown in Figure 4.2. Data from online forums are collected using a well-designed crawler. Different types of information are extracted from HTML pages and stored in the forum database ranging from user profiles to their posts. An inverted index is created to enable fast search of the forum posts and other information given a specific query term, and the index is updated periodically with the data differential from the forum database. When a user submits a query, it is first sent to the index engine, and two basic searching mechanisms, Okapi BM25 and the language model, are applied here to produce a merged set of candidate threads. Next, a set of features are extracted from each of the candidate threads and consumed by the authoritative ranking model to generate the final result threads.

4.3.2 Features

Features are very important to a machine learning algorithm. The decision by users to adopt a certain piece of information is determined by their perception of that information's utility. According to an information adoption model (*Sussman and Siegal*, 2003; *Rabjohn et al.*, 2008; *Wang et al.*, 2009), source credibility and argument quality are two key components of information usefulness. Inspired by this model, we propose a set of features, which can be grouped into two categories: argument quality (AQ) features and source credibility (SC) features. In the remainder of this section, we talk about these two categories of features in detail.

Argument quality features

Argument quality features are essentially thread-centric features, which are used to measure the informational quality and persuasive force of a thread itself. Different types of AQ features have been discussed a lot in previous studies (*Rabjohn et al.*, 2008; *Bailey and Pearson*, 1983). We compile them into four dimensions: relevance, information richness, timeliness, and completeness–as we did in previous research (*Wang et al.*, 2009). Table 4.3 lists the features.

Table 4.3: Argument quality features (the first column contains the abbreviations of features)

Relevance	
osim	Okapi BM25 similarity–Okapi BM25-based query-document relevance measure
lsim	Language model similarity–Language model-based query-document relevance measure
Information Richness	
len	Thread length–Number of words
nop	Number of posts–Number of posts in a thread
noa	Number of attachments–Number of images, links, code snippets and attachments
Timeliness	
pfq	Posting frequency–Number of posts per hour in a thread
hur	Thread duration–Number of hours elapsed between the first post and the last one
Completeness	
nou	Distinct opinions–Number of distinct members in a thread

Okapi BM25 is a ranking function-based probability retrieval framework (*Robertson et al.*, 1996) and is adopted to measure the relevance between documents and user queries. The

equation for calculating Okapi BM25 is given as follows:

$$s(d,q) = \sum_{i=1}^{n} \text{idf}(t_i) \frac{\text{tf}(t_i,d)(k_1+1)}{\text{tf}(t_i,d) + k_1(1-b+b\frac{|d|}{avgdl})}$$

where tf is the term-frequency of t_i in document d, $|d|$ is the number of terms, avgdl is the average document length, and k_1 and b are parameters, normally given as 2 and 0.5. IDF is short for inverse document frequency and the component is defined as

$$\text{idf}(t_i) = \log \frac{N - \text{df}(t_i) + 0.5}{\text{df}(t_i) + 0.5}$$

where df is the document-frequency of term t_i and N is the total number of documents in the collection. A higher IDF value means that the term is less common in the entire corpus globally and deserves to carry higher weight where it appears.

A language model is widely used in many natural language processing applications (*Ponte and Croft*, 1998) to calculate the probability that a document's language model would generate a query.

$$P(q|d) = \prod_{i=1}^{n}((1-\lambda)P(t_i|d) + \lambda P_D(t_i))$$

where λ is a linear interpolation smoothing parameter and is given as 0.5 here. P_D is the global generative probability, which is defined as

$$P_D(t_i) = \frac{\sum \text{tf}(t_i,d)}{\text{cs}}$$

where cs is the total number of terms in the entire document collection.

Weerkamp and De Rijke (*Weerkamp and De Rijke*, 2008) found that the length of a blog post and the number of comments were good indicators of information quality for a blog. Experimental results demonstrated that retrieval effectiveness was significantly improved with the addition of these features. According to (*Hearst and Dumais*, 2009), multi-author blogs are often ranked higher than single- or dual-author blogs in terms of quality. Blog posts and forum threads are both online user-generated content; therefore good quality features used in post ranking should also be used to rank threads. Similarly, features in the dimensions of timeliness and completeness should also be effective features indicating thread quality.

Source credibility features

Source credibility is the believability and authority of users who contribute to forum threads and is an aggregation of user metrics at the thread level. In (*Fogg and Tseng*, 1999), Tseng and Fogg identified four types of source credibility:

- experienced credibility, e.g., trustworthiness built based on interacting with people over time
- reputed credibility, e.g., what third parties have reported
- surface credibility, e.g., credibility arising from the way someone dresses or uses language
- presumed credibility, e.g., believing that your friends tell the truth

Specific to the task of thread ranking, two groups of source credibility features are developed (as listed in Table 4.4).

Table 4.4: Source credibility features (weighted averaged features are prefixed by "w")

Perceived Expertise	
(w)ady	Active membership length–Number of days elapsed between first and last posting dates
(w)anp	Productivity–Number of posts per year
(w)anf	Knowledge scope–Number of forums participated in by the member
Perceived Trustworthiness	
(w)acl	Closeness–Communication distance of the member from all other members
(w)abt	Betweenness–Importance in forming a coherent communication network
(w)aod	Out-degree–Number of people replied to by the member (i.e., generosity)
(w)aid	In-degree–Number of people who helped the member (i.e., popularity)
(w)acc	Clustering coefficient–How well connected the neighbors of the member are (i.e., brokerage)

The first group of features is relatively easy to derive, and the equations to compute the perceived knowledge features are listed as follows(*Wasserman and Faust*, 1994):

- In-degree is normalized by the number of vertices in the graph. The equation is given as follows:

$$c_i(v) = \frac{\deg^-(v)}{g-1}$$

  where g is the number of vertices in the graph and $\deg^-$ means indegree.

- Out-degree is also a normalized version. The equation is given as follows:

$$c_i(v) = \frac{\deg^+(v)}{g-1}$$

  where $\deg^+$ means outdegree.

- Clustering coefficient is a measurement of the degree to which a graph tends to group together. Given $N_i = \{v_j : e_{ij} \in E \wedge e_{ji} \in E\}$, a cluster coefficient is defined as

$$\text{cc}(v) = \frac{|\{e_{jk}\}|}{\deg * (\deg - 1)}$$

  where $v_j, v_k \in N_i$ and $e_{jk} \in E$

- Closeness is defined as the length of the shortest path between two graph nodes. The equation is given as follows:

$$c_c(v) = \frac{n_v^2}{(g-1)\sum_{t \in V \setminus v} d_G(v,t)}$$

  where n_v is the number of nodes that are reachable by v, g is the total number nodes in the graph, and d_G is defined as the length of the shortest distance between two nodes.

- Betweenness is another centrality measurement defined as the number of shortest paths from all vertices to all others that pass through that node. The equation is given as follows:

$$c_b(v) = \frac{1}{(g-1)(g-2)} \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

  where σ_{st} is the number of shortest paths from s to t, and $\sigma_{st}(v)$ is the number of shortest paths through v. The computation of both closeness and betweenness is very expensive. The algorithm requires $O(|V| + |E|)$ space and runs in $O(|V||E|)$ time for a sparse and unweighted graph.

  The SC features listed in Table 4.4 are defined and computed at user level, whereas ranking is performed at thread level. To derive thread-level features, the per-user SC scores are aggregated in the following two ways:

  1. Uniform averaging over all the users who participated in the thread with uniform weight on each user
  2. Weighted averaging over all the users who participated in the thread with users weighted based on the number of posts they made

4.3.3 Genetic algorithm and genetic programming

In artificial intelligence, GA and GP are two heuristics for optimization and solution search that belong to a class of evolution algorithms. In a GA method, a population of strings or number vectors is evolving through inheritance, mutation, selection, and crossover to yield

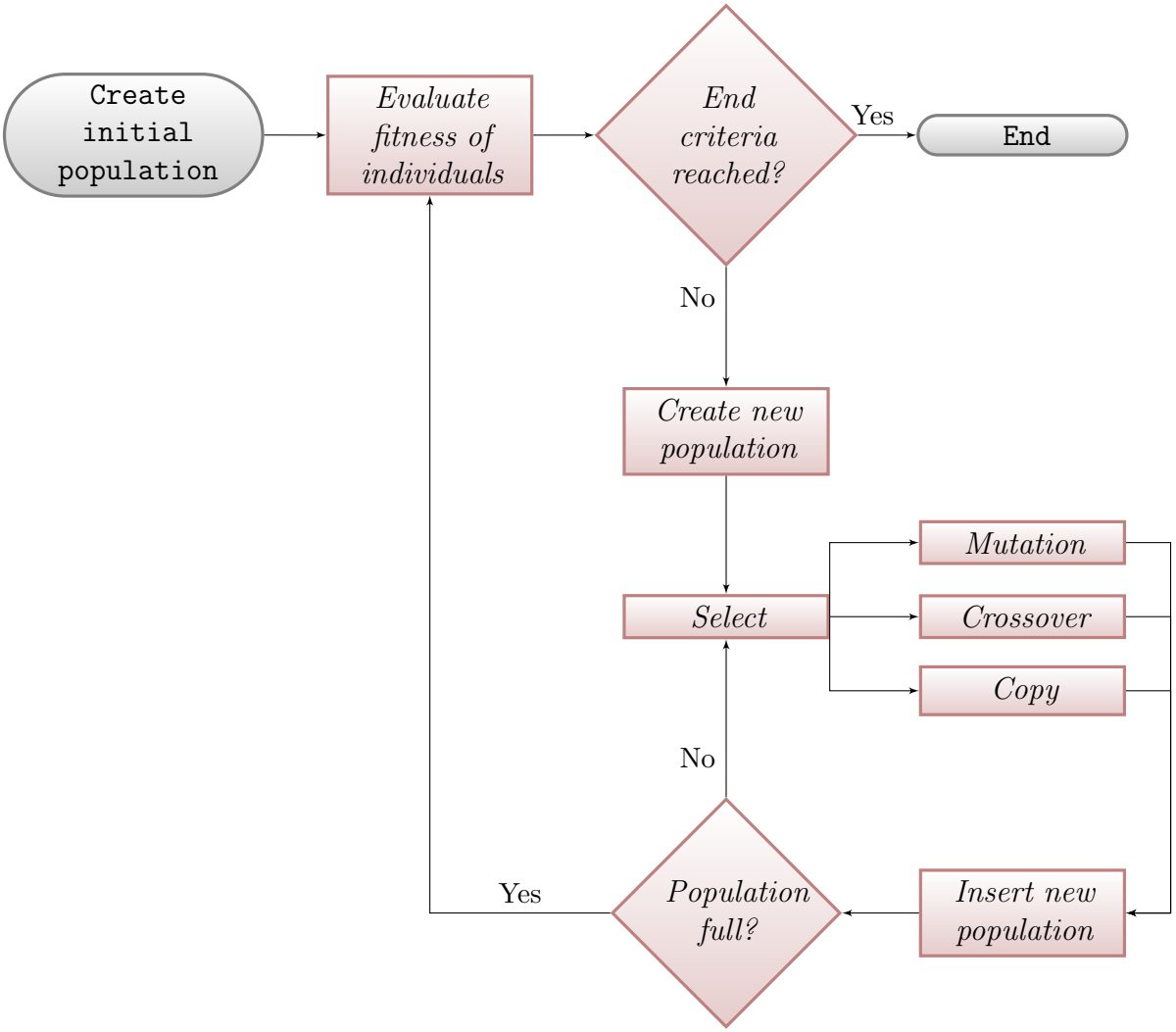

Figure 4.3: Flowchart of a genetic algorithm

better solutions in manner akin to natural evolution until the best solution is found or certain stop criteria are met. Figure 4.3 shows a typical flowchart of GA.

GP is different from GA in that each individual is a computer program usually represented as a tree of operators. Figure 4.4 illustrates a tree structure of function $f_1 - f_2 \cdot f_3 + \frac{2}{\log f_4}$.

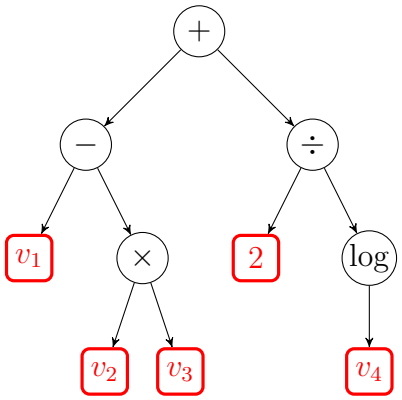

Figure 4.4: A GP function represented as a tree structure

4.3.4 Fitness functions

Both GA and GP require a fitness function to be constructed so as to rank one particular chromosome against the others and assess the optimality of a solution. The order-based relevance measurements, MAP and DCG (discounted cumulative gain), can be naturally adopted to meet the need. (We discuss in detail the measurements in Section 4.4.2.) Besides them, some self-defined fitness functions also proved effective (*Fan et al.*, 2004a). We know that documents with a higher authoritative level should be ranked at a higher position, so the fitness functions must be monotonically decreasing. Below, we list the fitness functions that are employed in GA and GP. Given a forum thread at position i, the fitness value can be calculated using the following three equations:

$$F_1(i) = \frac{1}{i}$$

$$F_2(i) = \frac{1}{\log(i)}$$

$$F_3(i) = c^i$$

where $0 < c < 1$ (note that DCG is actually a variation of F_2 and MAP is just F_1 with certain coefficients). Along with DCG, the charts for all the fitness functions are shown in the following figure.

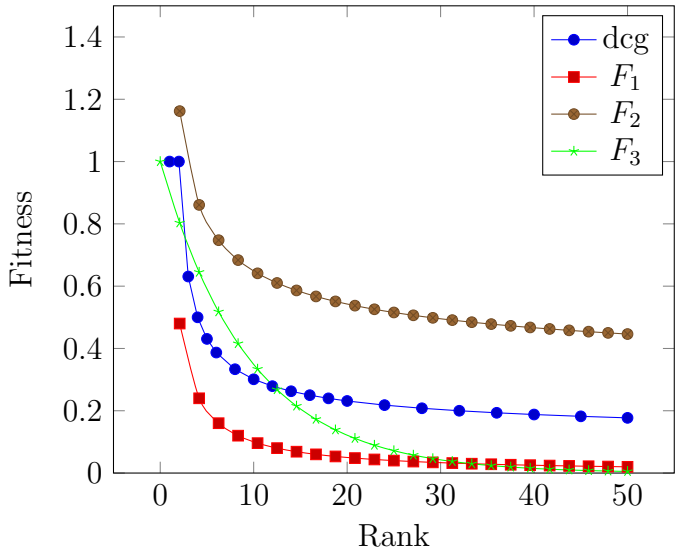

Figure 4.5: Fitness function

4.4 Experimental results

4.4.1 Data sets

We collected two data sets: vbCity and iPhone. vbCity is an online community for VB and .NET developers, and iPhone is one of the Apple communities focusing on iPhone usage. We list some dimensions of the forums in Table 4.5.

Forums	No. of sub-forums	No. of threads	No. of posts	No. of users
vbCity	33	157,730	662,897	37,612
iPhone	16	150,346	778,205	157,860

Table 4.5: Data sets

We randomly selected 50 threads from another online community in the same field, MSDN forums, using a filter that the selected threads must have at least 5 replies. We also removed those threads with general discussions and only preserved those aiming at resolving a specific user-encountered problem or question. From each of the selected threads, a query was manually generated in the form of a set of keywords by summarizing the key problem or question. The derived queries are very specific and discriminative with an average of 10 terms contained in each, so judges can easily navigate to the desired information by just looking at the queries. A sample set of the generated queries are listed in Table 4.6.

These queries were then fed into two search engines built by the vbCity community with Okapi and LM separately, and the top result threads from the two ranking functions were collected and merged to form the evaluation thread set. We selected the result threads from

Table 4.6: vbCity forum

1	How can I call a user defined SQL function in VB
2	How to automatically get the width and height of a JPEG file in VB
3	How to communicate between two computers using sockets in VB
4	How to create a label which has text with subscripts and superscripts in VB
5	How to delete a file while it is running

the two different sources equally to make sure that at least 50 threads were collected for each query.

Two independent graduate students in computer science were employed to read and tag the query-thread pairs separately using the four ASLs as guidelines and then conferred to resolve any conflicts. Finally, we collected 48 queries with at least one related thread in vbCity, which were used as a golden standard for evaluating our authoritative searching.

4.4.2 Measurements

The measurements we used for evaluation are discussed in this section. In traditional information retrieval, each document has only a binary judgment value: 1 or 0, which corresponds to whether the document is relevant to the query or not. Precision at position n ($P@n$) is defined as the ratio of relevant documents in the top n results. The average precision (AP) is defined below:

$$\mathrm{AP@}n = \frac{\sum_{i=1}^{n} P@i \cdot \mathrm{rel}(d_i)}{\sum_{i=1}^{N} \mathrm{rel}(d_i)} \tag{4.1}$$

where N is the size of the document corpus. Mean average precision (MAP) is calculated as the mean value of AP over the test query set.

$$\mathrm{RR@}n = \begin{cases} \dfrac{1}{n} & \text{if } \exists i < k : \mathrm{rel}(d_i) = 1 \wedge \forall j < i : \mathrm{rel}(d_j) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

Similarly, MRR is the mean value of RR. While P, AP, and RR are mainly designed to measure the binary judgment ranking problem, DCG can be leveraged to measure the cases with multiple scale levels. Given the human ratings of documents at different positions, DCG is defined as

$$\mathrm{DCG@}n = \mathrm{rate}(1) + \sum_{i=2}^{n} \frac{\mathrm{rate}(d_i)}{\log_2(i)} \tag{4.3}$$

To normalize DCG values, another measure, namely normalized DCG (NDCG), is given as DCG divided by the maximum value of it, which is DCG corresponding to the ideal ranking of all the documents (from the largest scale level to the smallest).

$$\text{NDCG@}n = \frac{\text{DCG@n}}{\text{IDCG@n}} \tag{4.4}$$

where IDCG indicates the maximum DCG. It is clear that the range of NDCG is $[0, 1]$.

4.4.3 Experimental design

Baseline methods

The baseline methods we used in experiments include two ranking functions: Okapi BM25 (*Robertson*, 1997) and the language model (LM) (*Ponte and Croft*, 1998), and one learning method: support vector machine (SVM) (*Hearst et al.*, 1998). It needs to be noted that we only used content features in the SVM method, which contains all the terms appearing in the threads.

Dataset partition strategies

GA and GP are essentially machine learning heuristics, which require a training dataset and a testing dataset. We designed two experiments to investigate the performance of GA and GP from different angles. The first experiment trained and tested the framework at the individual query level (thread set partition) with the aim of personalized ranking, while the second experiment trained and tested at the query set level (query set partition) with the aim of consensus ranking or generic ranking (*Fan et al.*, 2004b). For both the experiments, we used Okapi and LM baseline methods, but due to the nature of SVM, we only ran it in the first experiment.

Validation data set

We followed the three-data set design of (*Mitchell*, 1997; *Fan et al.*, 2004c,a) and randomly split the entire data set into three subsets: training (50%), validation (20%) and testing (30%) using the aforementioned two strategies. The intent for the introduction of the validation data set was to alleviate the problem of over-fitting on the training data set with GA and GP and to get a generalizable feature weight set or ranking function.

Fitness functions

Below we added some parameters to the self-defined fitness functions, and we improved the ranking function through parameter tuning.

$$F_1(i) = \left(\frac{1}{i + k_1} + k_2\right)^{k_3} \tag{4.5}$$

$$F_2(i) = \frac{1}{\log(i + k_4) + k_5} \tag{4.6}$$

$$F_3(i) = k_6^i \tag{4.7}$$

where ks in the formulas serve as parameters. Without losing the monotonically decreasing property of these fitness functions, the search spaces for all of the parameters are given as $k_1 \in [0, 5]$, $k_2 \in [0, 1]$, $k_3 \in [0.5, 5]$, $k_4 \in [0.2, 2]$, $k_5 \in [0, 1]$ and $k_6 \in [0.55, 0.9]$.

Global and local source credibility features

As discussed in the methodology section, there are features such as closeness and betweenness in the source credibility feature group. Typically, a forum contains hundreds or thousands of users and about the same number of connections between users. To compute these centrality measures of each individual node in the global graph is very expensive. Actually, the problem we are resolving is a search problem, and only a small portion of threads are relevant threads. Therefore, it is not necessary to compute user centralities based on the entire graph. Furthermore, constraining the scope to a small number of people who really have expertise in the search domain might produce more precise results. Considering this, in the experiments we also attempted to compute source credibility features based on the users participating in the top 1% of result threads of Okapi BM25, and we call these features local source credibility features to separate them from global source credibility features.

GA and GP model setting

The settings of a GA process include the following parts: genome, fitness functions, crossover, and mutation. Details are listed in Table 4.7.

Table 4.7: GA model settings

Component	Value
Genome	8 AQ features and 2 × 8 SC features (Tables 4.3,4.4)
Fitness	DCG, F_1, F_2, and F_3
Crossover	1.0
Mutation	0.05

The configuration of a GP system includes settings for the following components: terminals, functions, fitness functions, reproduction, crossover, and mutation. Table 4.8 lists the GP settings.

Table 4.8: GP model settings

Component	Value
Terminals	8 AQ features, 2 × 8 SC features (Tables 4.3,4.4), and constants (2 and 0.5)
Functions	+, −, ×, ÷, and log
Fitness	DCG, F_1, F_2, and F_3
Reproduction	0.85
Crossover	0.1
Mutation	0.05

4.4.4 Overall results

Below, we use MAP@10 as the major measure, which means that if two methods both have one measure better than the other, we will use MAP@10 as the standard. The percentage in bold denotes that the improvement is statistically significant; $^{\flat}$ indicates that the numbers are generated using NDCG as the fitness function; ^{n} indicates that numbers are based on new fitness functions.

vbCity

Table 4.9: Results of vbCity test set by *thread* partition with full feature set (global source credibility features) using relevance threshold larger or equal to ASL1

	MAP@10	NDCG@10	P@10	MRR@10
Okapi	0.1459	0.5963	0.3857	0.7344
LM	0.121	0.5819	0.3	0.6139
SVM	0.1313	0.5181	0.3667	0.6181
GA$^{\flat}$	0.1502	0.6142	0.3905	0.7325
GP$^{\flat}$	0.1522	0.6205	0.3171	0.6340
GA^{1}	0.1518	0.6481	0.3881	0.7812
GP^{1}	0.1562	0.6649	0.4048	0.7463
GA^{2}	0.1674	0.6685	0.3537	0.7289
GP^{2}	0.1540	0.6379	0.4048	0.7490
GA^{3}	0.1437	0.6151	0.3905	0.7253
GP^{3}	0.1468	0.6268	0.3929	0.7399

According to the results in Table 4.9, Okapi performed the best among three baseline methods with default parameter settings, and the language model was not as good as the other two.

Using NDCG as the fitness function, GP and GA already demonstrated some potential compared to the baseline methods. With new fitness methods $F_2()$ and $F_2()$, the results were even better.

Table 4.10: Results of vbCity test set by *query* partition with full feature set (global source credibility features) using relevance threshold larger or equal to ASL1

	MAP@10	NDCG@10	P@10	MRR@10
Okapi	0.2822	0.4956	0.5467	0.8083
LM	0.2742	0.4681	0.5267	0.8317
$GA^\flat$	0.2704	0.5229	0.5133	0.8541
$GP^\flat$	0.3033	0.4736	0.5667	0.7528
GA^1	0.2693	0.5188	0.5333	0.8233
GP^1	0.3004	0.5122	0.5467	0.8400
GA^2	0.3037	0.5567	0.4600	0.7833
GP^2	0.2947	0.5074	0.5533	0.7733
GA^3	0.2246	0.4587	0.4667	0.7583
GP^3	0.2904	0.4969	0.5467	0.8095

Table 4.10 shows the results on the query partition, which are much better compared to the thread partition results. This is because that in the thread partition, threads of different ASLs are uniformly distributed in each partition, and the test set only contained a certain portion of related threads. In the query partition, each query in the test set has all the related threads. Similar to the thread partition results, Okapi outperformed the language model, and GA and GP displayed some gains. To summarize the results at a relevance threshold larger or equal to ASL1, the proposed new features did not show too much advantage when the goal was only to search for relevant threads, with a maximum 7% gain on MAP.

Table 4.11: Results of vbCity test set by *thread* partition with full feature set (global source credibility features) using relevance threshold larger or equal to **ASL2**

	MAP@10	NDCG@10	P@10	MRR@10
Okapi	0.1378	0.6021	0.3146	0.6536
LM	0.1364	0.5918	0.3714	0.7145
SVM	0.1164	0.4966	0.2854	0.5007
$GA^\flat$	0.1440	0.6169	0.3905	0.7346
$GP^\flat$	0.1428	0.5971	0.3293	0.6803
GA^1	0.1629	0.6299	0.3463	0.7123
GP^1	**0.1642**	**0.6566**	**0.3488**	**0.7547**
GA^2	0.1591	0.6353	0.3366	0.7187
GP^2	0.1633	0.6508	0.3512	0.7218
GA^3	0.1554	0.6324	0.3341	0.7082
GP^3	0.1525	0.6298	0.3463	0.6815

Table 4.12: Results of vbCity test set by *query* partition with full feature set (global source credibility features) using relevance threshold larger or equal to **ASL2**

	MAP@10	**NDCG@10**	**P@10**	**MRR@10**
Okapi	0.2647	0.4956	0.3933	0.7639
LM	0.2296	0.4681	0.3467	0.7006
$GA^{\flat}$	0.2898	0.5229	0.4333	0.8207
$GP^{\flat}$	0.2710	0.4736	0.4333	0.6861
GA^{1}	0.2892	0.5234	0.4400	0.8194
GP^{1}	**0.2984**	**0.5141**	**0.4467**	**0.7956**
GA^{2}	0.2772	0.5159	0.4200	0.7639
GP^{2}	0.2968	0.5229	0.4400	0.8095
GA^{3}	0.2425	0.4495	0.3867	0.7241
GP^{3}	0.2818	0.4981	0.4133	0.8241

According to the results in Tables 4.11 and 4.12, the proposed new features were very powerful when it came to authoritative thread searching ($\geq$ ASL2). The gains over the Okapi method were 19.12% and 12.74% for thread partition and query partition, respectively. GP results were better than GA, and new fitness functions were better than NDCG. The computation of social centrality features is very expensive. As is discussed below, the experiments were repeated, where global centrality features were substituted by a set of local centrality features.

Table 4.13: Results of vbCity test set by *thread* partition with full feature set (**local** source credibility features) using relevance threshold larger or equal to ASL1

	MAP@10	**NDCG@10**	**P@10**	**MRR@10**
Okapi	0.1459	0.5963	0.3857	0.7344
$GA^{\flat}$	0.1578	0.6551	0.4119	0.7321
$GP^{\flat}$	0.1515	0.6403	0.4000	0.7589
GA^{1}	0.1885	0.7238	0.5920	0.9148
GP^{1}	0.1906	0.7226	0.5860	0.9174
GA^{2}	0.1620	0.6743	0.4048	0.7774
GP^{2}	0.1539	0.6333	0.4048	0.7585
GA^{3}	0.1654	0.6840	0.4000	0.7984
GP^{3}	0.1515	0.6472	0.4048	0.7677

As we can see from Tables 4.13, 4.14, 4.15, 4.16, local centrality features almost covered the information in global centrality features, and most results using argument quality features plus local source credibility features were comparable or even slightly higher in certain cases. GP and GA still outperformed baseline methods, and new fitness functions were better than DCG.

Following are the GA weights and GP formula for the rows in bold in Tables 4.15 and 4.16,

Table 4.14: Results of vbCity test set by *query* partition with full feature set (**local** source credibility features) using relevance threshold larger or equal to ASL1

	MAP@10	**NDCG@10**	**P@10**	**MRR@10**
Okapi	0.2822	0.4956	0.5467	0.8083
$GA^\flat$	0.2770	0.5240	0.5267	0.8083
$GP^\flat$	0.2583	0.4935	0.4933	0.8095
GA^1	0.2331	0.5713	0.6333	0.9111
GP^1	0.2556	0.6219	0.6533	0.9444
GA^2	0.3018	0.5567	0.5667	0.8500
GP^2	0.2921	0.5269	0.5600	0.8095
GA^3	0.2608	0.5039	0.5333	0.8167
GP^3	0.2998	0.5006	0.5600	0.8429

Table 4.15: Results of vbCity test set by *thread* partition with full feature set (**local** source credibility features) using relevance threshold larger or equal to **ASL2**

	MAP@10	**NDCG@10**	**P@10**	**MRR@10**
Baseline	0.1378	0.6021	0.3146	0.6536
$GA^\flat$	0.1585	0.6402	0.3512	0.6957
$GP^\flat$	0.1519	0.6179	0.3463	0.6416
GA^1	**0.1704**	**0.6636**	**0.3463**	**0.7389**
GP^1	0.1657	0.6491	0.3585	0.6997
GA^2	0.1674	0.6685	0.3537	0.7289
GP^2	0.1612	0.6368	0.3463	0.7153
GA^3	0.1650	0.6579	0.3488	0.7226
GP^3	0.1610	0.6491	0.3512	0.6689

Table 4.16: Results of vbCity test set by *query* partition with full feature set (**local** source credibility features) using relevance threshold larger or equal to **ASL2**

	MAP@10	**NDCG@10**	**P@10**	**MRR@10**
Baseline	0.2647	0.4956	0.3933	0.7639
$GA^\flat$	0.2871	0.5311	0.4533	0.7333
$GP^\flat$	0.2764	0.4935	0.4133	0.7762
GA^1	0.3001	0.5446	0.4400	0.8222
GP^1	**0.3058**	**0.5344**	**0.4333**	**0.8800**
GA^2	0.3037	0.5567	0.4600	0.7833
GP^2	0.2954	0.5323	0.4333	0.8095
GA^3	0.2967	0.5396	0.4133	0.8333
GP^3	0.2880	0.5211	0.4400	0.8407

respectively.

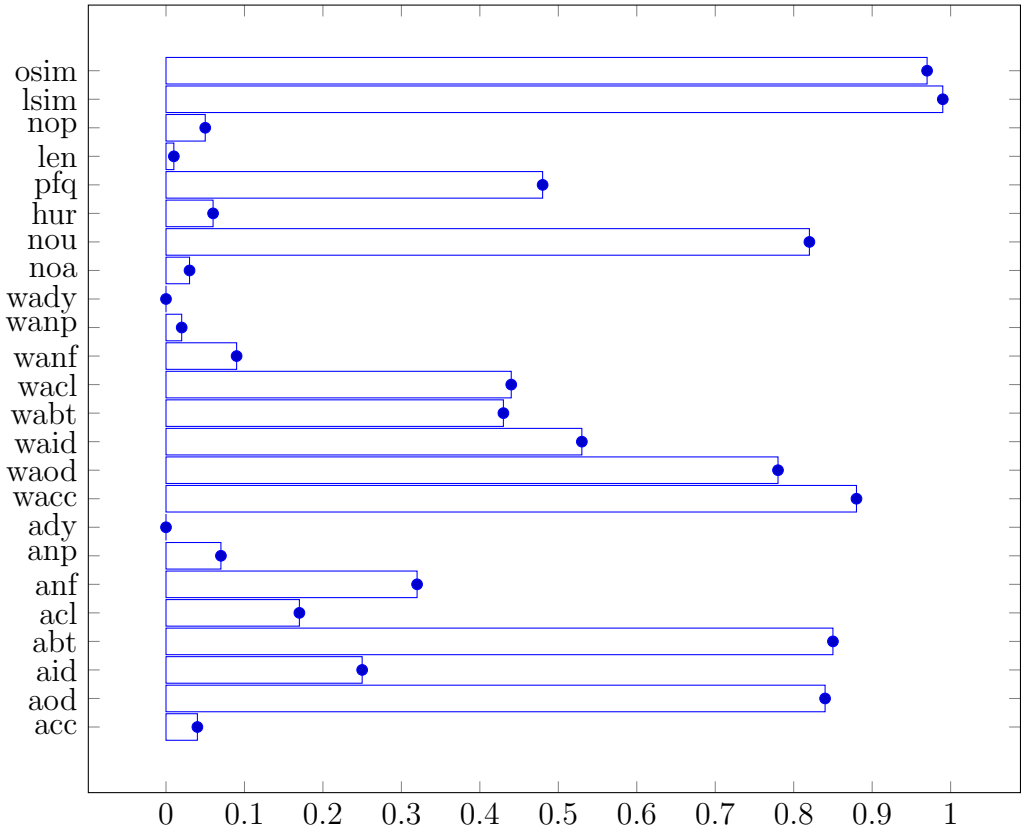

Figure 4.6: Feature weights in GA

$$\begin{aligned}\text{score} =& 3 \cdot \text{lsim} + \text{nou} + 2.5 \cdot \text{osim} + \text{waod} + \log \frac{\text{noa} + \text{wanp} + \text{nop} \cdot \log(\text{wabt})}{\text{nop}} \\ &+ \log \left(\frac{\text{abt} + \text{pfq}}{\log(\text{wabt})} + \frac{\text{lsim} + \text{waod} + \text{anf}}{\text{aod}} \right)\end{aligned} \tag{4.8}$$

The terms in the table and the equation are short for the features listed in Tables 4.3 and 4.4. The terms lsim, osim and nou are all AQ features, and they play a very important role in both GA and GP, where lsim and osim are relevance features and nou means number of users. On the other hand, the SC features including acc, abt, and aod carry high weight in GA and are also very prominent in the ranking formula. Therefore, both AQ and SC features proved to be effective in thread ranking.

iPhone

To further test our proposed learning to rank method, experiments were conducted under the same settings on the iPhone data set and the results are presented below.

Table 4.17: Results of iPhone test set by *thread* partition with full feature set (**local** source credibility features) using relevance threshold larger or equal to ASL1

	MAP@10	**NDCG@10**	**P@10**	**MRR@10**
Baseline	0.1459	0.5963	0.3857	0.7344
$GA^{\flat}$	0.1771	0.7067	0.5800	0.8697
$GP^{\flat}$	0.1829	0.7075	0.5820	0.8654
GA^{1}	0.1636	0.6864	0.4024	0.7817
GP^{1}	0.1544	0.6400	0.4048	0.7421
GA^{2}	0.1866	0.7182	0.5960	0.9123
GP^{2}	0.1851	0.7073	0.5880	0.9206
GA^{3}	0.1839	0.7165	0.5860	0.8940
GP^{3}	0.1839	0.7080	0.5840	0.8651

Table 4.18: Results of iPhone test set by *query* partition with full feature set (**local** source credibility features) using relevance threshold larger or equal to ASL1

	MAP@10	**NDCG@10**	**P@10**	**MRR@10**
Baseline	0.2822	0.4956	0.5467	0.8083
$GA^{\flat}$	0.2154	0.5441	0.5933	0.8556
$GP^{\flat}$	0.2322	0.5909	0.6067	0.9333
GA^{1}	0.3047	0.5513	0.5533	0.8667
GP^{1}	0.2998	0.5006	0.5600	0.8429
GA^{2}	0.2359	0.5846	0.6333	0.9333
GP^{2}	0.2475	0.6223	0.6400	0.9667
GA^{3}	0.2309	0.5744	0.6133	0.9333
GP^{3}	0.2314	0.5952	0.6133	0.9333

According to Tables 4.17, 4.18, 4.19 and 4.20, very similar results were obtained for the iPhone data set. The learning to rank method was very effective in thread ranking. Different from vbCity, GP outperformed GA in almost every setting. There are two rows of results marked with † in Tables 4.19 and 4.20, where we applied the GA weights and GP formula obtained from the vbCity data set (Tables 4.15 and 4.16). As we can see, although they are not as good as the optimal weights or formula for the iPhone data set, the results are still very promising. Therefore both the weights and formula are generalizable.

Table 4.19: Results of iPhone test set by *thread* partition with full feature set (**local** source credibility features) using relevance threshold larger or equal to **ASL2**

	MAP@10	**NDCG@10**	**P@10**	**MRR@10**
Baseline	0.1260	0.7477	0.3457	0.6691
$GA^{\flat}$	0.1906	0.7067	0.4100	0.7537
$GP^{\flat}$	0.1950	0.7220	0.4140	0.7576
GA^{1}	0.1949	0.7185	0.4120	0.7803
GP^{1}	**0.2062**	**0.7252**	**0.4180**	**0.8094**
GA^{2}	0.1956	0.7285	0.4140	0.8279
GP^{2}	0.2028	0.7094	0.4200	0.8119
GA^{3}	0.1900	0.7020	0.4140	0.7600
GP^{3}	0.2001	0.7286	0.4060	0.8328
$GA^{\dagger}$	0.1740	0.6684	0.398	0.7103

Table 4.20: Results of iPhone test set by *query* partition with full feature set (**local** source credibility features) using relevance threshold larger or equal to **ASL2**

	MAP@10	**NDCG@10**	**P@10**	**MRR@10**
Baseline	0.1638	0.4387	0.3667	0.6639
$GA^{\flat}$	0.2643	0.5441	0.4867	0.8556
$GP^{\flat}$	0.2782	0.5927	0.4867	0.8667
GA^{1}	0.2970	0.5827	0.5400	0.9222
GP^{1}	**0.3321**	**0.6219**	**0.5533**	**0.9444**
GA^{2}	0.2936	0.5846	0.5400	0.9333
GP^{2}	0.3237	0.6100	0.5333	0.9444
GA^{3}	0.2741	0.5572	0.5200	0.8222
GP^{3}	0.3044	0.5997	0.5400	0.9222
$GP^{\dagger}$	0.2544	0.5651	0.4733	0.8388

4.5 Conclusion

In this chapter, an effective learning to rank method was proposed based on genetic evolution technologies to solve the thread ranking problem in online forums. An enriched feature set including argument quality features and source credibility features was developed, and the two groups of features both proved to be very effective in tests. GA and GP approaches performed excellently in finding local optimal solutions, and GP performed better than GA on the iPhone forum data. The well-devised fitness function greatly improved efficiency. Experimental results showed that the authoritative ranking method could effectively find threads with both high relevance and high authoritativeness that could best serve users' information needs.

In the future, the relationship between posts of a single thread is an interesting topic. We will look into how a thread is gradually expanded and how the solution to a problem or the answer to a question is drawn step by step with the help of community members. A hidden Markov model (HMM) could be potentially applied to model the process, and the post that eventually solves the user problem could be precisely located. In this way, we could potentially enable better answers to user questions in an online community.

Chapter 5

Conclusion and Future Work

This study focused specifically on product defects and presented a framework for finding and summarizing product defects, and ranking helpful threads. The framework consists of three major components: product defect detection, product feature extraction, and learning to rank threads. Each component targets at a challenging problem in the process of detecting, summarizing, and correcting product defects. Machine learning methods were adopted to address these problems and data from online forums was used to evaluate the effectiveness of the framework.

Product defects are a major concern for product manufactures and service providers. Traditionally, product defects are detected through a series of carefully devised tests and inspections. However, the entire process is very expensive and time consuming, and after a product enters the market, there is almost no reliable way to find defects. The first chapter of this dissertation proposed to utilize the data from online forums and address the problem of product defect detection using a supervised classification method. The classification framework is feature-friendly, adaptive, and generalizable. Experiments were conducted on different data sets, and testing results show that the proposed method is very effective in separating defect-related discussions from non-defect issues.

Chapter two proposed an effective method for summarizing defect-related threads using product features and feature clusters. Machine learning methods were adopted to approach both tasks: the classifier with the best performance was selected from four alternative methods and was then applied to separate product feature phrases from non-feature phrases; based on a combined similarity function, the DBSCAN clustering method was adopted to group the extracted features into feature clusters. The performance of both methods was evaluated based on human judgments and the experimental results are very promising.

To utilize the collective intelligence in online forums, a learning to rank method was adopted to find helpful threads for solving product-related problems. According to the knowledge adoption model, two types of features were developed for predicating helpful threads: argu-

ment quality features and source credibility features. Next, genetic algorithm and genetic programming methods were employed to find the optimal ranking solutions. Experiments were conducted based on data from two different forums. The results show that the features are good indicators for thread helpfulness, and both GA and GP methods are very effective across forums.

Looking forward, several research directions are identified based on the current work. First, system performance can be improved with feature refinement. For example, the current feature set for product defect detection only contains single features, however, according to (*Wang et al.*, 2012), combined features could greatly improve the performance. Second, the underlying logic relationship among thread posts is worth exploring. For example, thread ranking can benefit from understanding how a discussion is gradually expanded and how a problem solution is derived step by step. A hidden Markov model could potentially be applied to model this process. Third, more experiments will be conducted on forum data in different industries. Currently, the data used for product defect detection is from a single industry–automobile, and it would be interesting and valuable to examine the model performance on data from a different industry, e.g., electronics. In addition, five well-developed conference and journal papers have been already published based on this dissertation, and more publications are targeted to move the research forward.

Bibliography

Abbasi, A., and H. Chen (2005), Applying authorship analysis to extremist-group web forum messages, *Intelligent Systems, IEEE*, *20*(5), 67–75.

Abbasi, A., and H. Chen (2008), Writeprints: a stylometric approach to identity-level identification and similarity detection in cyberspace, *ACM Transactions on Information Systems (TOIS)*, *26*(2), 7.

Abbasi, A., H. Chen, and A. Salem (2008), Sentiment analysis in multiple languages: feature selection for opinion classification in web forums, *ACM Transactions on Information Systems (TOIS)*, *26*(3), 12.

Abrahams, A. S., J. Jiao, W. Fan, G. A. Wang, and Z. Zhang (2013, forthcoming), What's buzzing in the blizzard of buzz? automotive component isolation in social media postings, *Decision Support Systems*.

Alpaydin, E. (2010), *Introduction to machine learning*, 2nd edition ed., The MIT Press.

Bailey, J. E., and S. W. Pearson (1983), Development of a tool for measuring and analyzing computer user satisfaction, *Management science*, *29*(5), 530–545.

Baldwin, T., D. Martinez, and R. B. Penman (2007), Automatic thread classification for linux user forum information access, in *Proceedings of the 12th Australasian Document Computing Symposium*, pp. 72–79.

Beitzel, S. M. (2006), On understanding and classifying web queries, Ph.D. thesis, Illinois Institute of Technology.

Berry, M. J., and G. Linoff (1997), *Data mining techniques: for marketing, sales, and customer support*, John Wiley & Sons, Inc.

Bishop, C. M. (1995), *Neural Networks for Pattern Recognition*, Oxford University Press.

Blair-Goldensohn, S., K. Hannan, R. McDonald, T. Neylon, G. A. Reis, and J. Reynar (2008), Building a sentiment summarizer for local service reviews, in *WWW Workshop on NLP in the Information Explosion Era*.

Blei, D. M., A. Y. Ng, and M. I. Jordan (2003), Latent dirichlet allocation, *The Journal of Machine Learning Research*, *3*, 30.

Breiman, L., J. Friedman, and C. J. Stone (1984), *Classification and Regression Trees*, Chapman and Hal.

Bunkley, N. (2010), Toyota issues a 2nd recall, `http://www.nytimes.com/2010/01/22/business/22toyota.html`.

Burges, C., T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender (2005), Learning to rank using gradient descent, in *Proceedings of the 22nd International Conference on Machine learning*, ICML '05, pp. 89–96, ACM, New York, NY, USA.

Chang, C. C., and C. J. Lin (2011), Libsvm: a library for support vector machines, *ACM Transactions on Intelligent Systems and Technology (TIST)*, *2*(3), 27.

Chen, Y., and C. Lin (2006), Combining svms with various feature selection strategies, in *Studies in Fuzziness and Soft Computing*, pp. 315–324, Springer Berlin / Heidelberg.

Conroy, J. M., and D. P. O'leary (2001), Text summarization via hidden markov models, in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Corney, M., O. De Vel, A. Anderson, and G. Mohay (2002), Gender-preferential text mining of e-mail discourse, in *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*, pp. 282–289, IEEE.

Coussement, K., and D. Vandenpoel (2008), Improving customer complaint management by automatic email classification using linguistic style features as predictors, *Decision Support Systems*, *44*(4), 870–882.

Crammer, K., and Y. Singe (2001), Pranking with ranking, in *Advances in Neural Information Processing Systems 14*, pp. 641–647, MIT Press.

Dagan, I., Y. Karov, and D. Roth (1997), Mistake-driven learning in text categorization, in *Proceedings of the Second Conference on Empirical Methods in NLP*, pp. 55–63.

Das, D., and A. F. T. Martins (2007), A survey on automatic text summarization, *Literature Survey for the Language and Statistics II course at CMU*, *4*, 192–195.

Das, S. (2001), Filters, wrappers and a boosting-based hybrid for feature selection, in *Proceedings of the Eighteenth International Conference on Machine Learning*, pp. 74–81, Citeseer.

Dash, M., and H. Liu (1997), Feature selection for classification, *Intelligent Data Analysis*, *1*(3), 26.

Day, D., J. Aberdeen, L. Hirschman, R. Kozierok, P. Robinson, and M. Vilain (1997), Mixed-initiative development of language processing systems, in *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pp. 348–355, Association for Computational Linguistics.

De Vel, O. (2000), Mining e-mail authorship, in *Proceedings Workshop on Text Mining, ACM International Conference on Knowledge Discovery and Data Mining (KDD2000)*.

De Vel, O., A. Anderson, M. Corney, and G. M. Mohay (2001), Mining e-mail content for author identification forensics, in *SIGMOD Record*, vol. 30, pp. 55–64, ACM.

Dellarocas, C. (2006), Strategic manipulation of internet opinion forums: implications for consumers and firms, *Management Science*, *52*(10), 1577–1593.

Elsas, J. L., and J. G. Carbonell (2009), It pays to be picky: an evaluation of thread retrieval in online forums., in *SIGIR'09*, pp. 714–715.

Ester, M., H.-P. Kriegel, J. Sander, and X. Xu (1996), A density-based algorithm for discovering clusters in large spatial databases with noise, in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*.

Fan, W., E. A. Fox, P. Pathak, and H. Wu (2004a), The effects of fitness functions on genetic programming-based ranking discovery for web search, *Journal of the American Society for Information Science and Technology*, *55*(7), 628–636.

Fan, W., M. Gordon, and P. Pathak (2004b), A generic ranking function discovery framework by genetic programming for information retrieval, *Information Processing and Management*, *40*(4), 587–602.

Fan, W., M. D. Gordon, and P. Pathak (2004c), Discovery of context-specific ranking functions for effective information retrieval using genetic programming, *IEEE Trans. on Knowl. and Data Eng.*, *16*, 523–527.

Fan, W., M. D. Gordonb, and P. Pathak (2005), Effective profiling of consumer information retrieval needs: a unified framework and empirical comparison, *Decision Support Systems*, *40*(2), 21.

Fogg, B., and H. Tseng (1999), The elements of computer credibility, in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 80–87, ACM.

Freund, Y., R. Iyer, R. E. Schapire, and Y. Singer (2003), An efficient boosting algorithm for combining preferences, *The Journal of Machine Learning Research*, *4*, 933–969.

Fu, L., D. H.-L. Goh, and S. S.-B. Foo (2004), Query clustering using a hybrid query similarity measure, *WSEAS Transaction on Computers*, *3*(3), 700–705.

Gamon, M., A. Aue, S. Corston-Oliver, and E. Ringger (2005), Pulse: mining customer opinions from free text, *Advances in Intelligent Data Analysis VI*, pp. 121–132.

Griffiths, T. L., and M. Steyvers (2004), Finding scientific topics, in *Proceedings of the National Academy of Sciences*, vol. 101, pp. 5228–5235, National Acad Sciences.

Gunn, S. R. (1998), Support vector machines for classification and regression, *ISIS Technical Report 14*, University of Southampton.

Gusfield, D. (1997), *Algorithms on strings, trees, and sequences: computer science and computational biology*, Cambridge University Press.

Guyon, I., and A. Elisseeff (2003), An introduction to variable and feature selection, *The Journal of Machine Learning Research*, *3*, 26.

Han, J., T. Kim, and J. Choi (2007), Web document clustering by using automatic keyphrase extraction, in *Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops*, pp. 56–59, IEEE Computer Society.

Healey, J. R., and S. S. Carty (2010), Toyota recall: gas pedal issue affects more than 2m vehicles, `http://usatoday30.usatoday.com/money/autos/2010-01-21-toyota-recall-gas-pedal_N.htm`.

Hearst, M., S. Dumais, E. Osman, J. Platt, and B. Scholkopf (1998), Support vector machines, *Intelligent Systems and their Applications, IEEE*, *13*(4), 18–28.

Hearst, M. A., and S. T. Dumais (2009), Blogging together: an examination of group blogs, in *Proceedings of the International Conference on Weblogs and Social Media (ICWSM 2009)*.

Herbrich, R., T. Graepel, and K. Obermayer (1999), Large margin rank boundaries for ordinal regression, *Advances in Neural Information Processing Systems*, pp. 115–132.

Hofmann, T. (1999), Probabilistic latent semantic indexing, in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, p. 8.

Hosmer, D. W., and S. Lemeshow (2000), *Applied logistic regression*, 2nd edition ed., Wiley-Interscience, New York.

Hu, M., and B. Liu (2004a), Mining and summarizing customer reviews, in *Proceedings of the 10th ACM International Conference on Knowledge Discovery and Data Mining*.

Hu, M., and B. Liu (2004b), Mining opinion features in customer reviews, in *Proceedings of the 19th National Conference on Artificial Intelligence*.

Inmon, W. (1992), *Building the data warehouse*, John Wiley & Sons, Inc.

Isidore, C. (2010), Toyota recall costs: $2 billion, `http://www.forbes.com/2010/02/04/toyota-earnings-recall-markets-equities-prius.html`.

Jiang, X., Y. Hu, and H. Li (2009), A ranking approach to keyphrase extraction, in *Proceedings of the 32nd international ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 756–757, ACM.

Jones, S., and M. S. Staveley (1999), Phrasier: a system for interactive document retrieval using keyphrases, in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 160–167, ACM.

Kohavi, R., and G. H. John (1997), Wrappers for feature subset selection, *Artificial Intelligence*, *97*(1), 273–324.

Koppel, M., S. Argamon, and A. R. Shimoni (2002), Automatically categorizing written texts by author gender, *Literary and Linguistic Computing*, *17*(4), 401–412.

Kupiec, J., J. Pedersen, and F. Chen (1995), A trainable document summarizer, in *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, p. 6.

Lea, R. (2010), Toyota expects massive drop in sales figures, `http://www.allvoices.com/news/5324358/s/49352707-toyota-expects-massive-drop-in-sales-figures`.

Li, P., C. J. Burges, and Q. Wu (2007), Learning to rank using classification and gradient boosting, in *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)*, Citeseer.

Liautaud, B. (2000), *E-Business intelligence: turning information into knowledge into Profit*, McGraw-Hill, Inc.

Lin, C.-Y. (1999), Training a selection function for extraction, in *Proceedings of the 8th International Conference on Information and Knowledge Management*, p. 8.

Lippmann, R. (1987), An introduction to computing with neural nets, *ASSP Magazine, IEEE*, *4*(2), 4–22.

Liu, H., and L. Yu (2005), Toward integrating feature selection algorithms for classification and clustering, *Knowledge and Data Engineering, IEEE Transactions on*, *17*(4), 491–502.

Liu, T.-Y. (2009), Learning to rank for information retrieval, *Found. Trends Inf. Retr.*, *3*, 225–331.

Lynch, C. (2009), Ti's web 2.0 success story: better customer service, `http://www.cio.com/article/496338/TI_s_Web_2.0_Success_Story_Better_Customer_Service`.

Madigan, D., A. Genkin, D. D. Lewis, S. Argamon, D. Fradkin, and L. Ye (2005a), Author identification on the large scale, in *Joint Annual Meeting of the Interface and the Classification Society of North America (CSNA)*.

Madigan, D., A. Genkin, D. D. Lewis, and D. Fradkin (2005b), Bayesian multinomial logistic regression for author identification, *Bayesian Inference and Maximum Entropy Methods in Science and Engineering*, *803*, 509–516.

Miller, G., et al. (1995), Wordnet: a lexical database for english, *Communications of the ACM*, *38*(11), 39–41.

Mitchell, T. (1997), *Machine Learning*, Mcgraw Hill Higher Education.

Morville, P., and L. Rosenfeld (2006), *Information Architecture for the World Wide Web*, 1st edition ed., O'Reilly Media, Sebastopol, CA.

MotorTrend (2012), 2004 honda accord recalls & problems - motor trend magazine, `http://www.motortrend.com/cars/2004/honda/accord/recalls/`.

Ng, H. T., W. B. Goh, and K. L. Low (1997), Feature selection, perceptron learning, and a usability case study for text categorization, in *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, p. 7.

Nowson, S., and J. Oberlander (2006), The identity of bloggers: openness and gender in personal weblogs, in *Proceedings of the AAAI Spring Symposia on Computational Approaches to Analyzing Weblogs*.

NYTimes (2006), Battery recall could cost $400 million for dell and sony - business - international herald tribune, `http://www.nytimes.com/2006/08/15/business/worldbusiness/15iht-batteries.2493980.html`.

Okurowski, M. E., H. Wilson, J. Urbina, T. Taylor, R. C. Clark, and F. Krapcho (1999), A trainable summarizer with knowledge acquired from robust nlp techniques, *Advances in Automatic Text Summarization*, pp. 71–80.

Pang, B., L. Lee, and S. Vaithyanathan (2002), Thumbs up?: sentiment classification using machine learning techniques, in *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, vol. 10, pp. 79–86, Association for Computational Linguistics.

Ponte, J. M., and W. B. Croft (1998), A language modeling approach to information retrieval, in *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 275–281, ACM.

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

Quinlan, J. R. (1993), *C4. 5: programs for machine learning*, 1st edition ed., Morgan kaufmann, San Francisco, CA.

Rabjohn, N., C. M. K. Cheung, and M. K. O. Lee (2008), Examining the perceived credibility of online opinions: information adoption in the online environment, in *Hawaii International Conference on System Sciences*, vol. 0, p. 286, IEEE Computer Society.

Ramage, D., and E. Rosen (2009), Stanford topic modeling toolbox, `http://nlp.stanford.edu/software/tmt/tmt-0.3/`.

Rijsbergen, C. J. V. (1979), *Information Retrieval*, 2nd ed., Butterworth-Heinemann, Newton, MA, USA.

Rish, I. (2001), An empirical study of the naive bayes classifier, in *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, vol. 3, pp. 41–46.

Robertson, S. E. (1997), Overview of the okapi projects., *Journal of Documentation*, *53*(1).

Robertson, S. E., S. Walker, M. Beaulieu, M. Gatford, and A. Payne (1996), Okapi at TREC-4, in *Proceedings of the fourth text retrieval conference*, pp. 73–97, NIST Special Publication.

Schwartz, B. (2008), The google quality raters handbook, `http://searchengineland.com/the-google-quality-raters-handbook-13575`.

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

Seo, J., W. B. Croft, and D. A. Smith (2009), Online community search using thread structure, in *Proceeding of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pp. 1907–1910, ACM, New York, NY, USA.

Slater, S. F., and J. C. Narver (2000), Intelligence generation and superior customer value, *Journal of the Academy of Marketing Science*, *28*(1), 8.

Somol, P., J. Novovicová, and P. Pudil (2010), Efficient feature subset selection and subset size optimization, *Pattern Recognition Recent Advances*, pp. 1–24.

Stodder, D. (2010a), How text analytics drive customer insight, `http://www.informationweek.com/software/business-intelligence/how-text-analytics-drive-customer-insigh/222600270`.

Stodder, D. (2010b), Text analytics drives customer insight, `http://www.informationweek.com/software/business-intelligence/text-analytics-drives-customer-insight/222400433`.

Stone, P. J., R. F. Bales, J. Z. Namenwirth, and D. M. Ogilvie (1962), The General Inquirer: a computer system for content analysis and retrieval based on the sentence as a unit of information, *Behavioral Science*, *7*(4), 484–498.

Stone, P. J., D. C. Dunphy, and M. S. Smith (1966), The General Inquirer: a computer approach to content analysis, *MIT Press*.

Sussman, S. W., and W. S. Siegal (2003), Informational influence in organizations: an integrated approach to knowledge adoption, *Information Systems Research*, *14*(1), 47–65.

Tabuchi, H., and M. Maynard (2009), President of Toyota apologizes, `http://www.nytimes.com/2009/10/03/business/global/03toyota.html`.

Titov, I., and R. McDonald (2008), Modeling online reviews with multi-grain topic models, in *Proceeding of the 17th International Conference on World Wide Web*, pp. 111–120, ACM.

Valdes-Dapena, P. (2010), NHTSA faces tough questions on Toyota, `http://dealerdigestdaily.com/2010/02/02-23-10.html`.

Valizadegan, H., R. Jin, R. Zhang, and J. Mao (2010), Learning to rank by optimizing ndcg measure, in *Neural Information Processing Systems*, vol. 22, pp. 1883–1891, Citeseer.

Vance, A. (2006), Dell recalls 4.1m laptop batteries, `http://www.theregister.co.uk/2006/08/14/dell_laptop_recall/`.

Wang, G. G., J. Jiao, and W. Fan (2009), Searching for authoritative documents in knowledge-base communities, in *ICIS 2009 Proceedings*.

Wang, Y., K. Han, and D. Wang (2012), Exploring monaural features for classification-based speech segregation, *21*(2).

Wasserman, S., and K. Faust (1994), *Social network analysis: methods and applications*, vol. 8, Cambridge University Press.

Weerkamp, W., and M. De Rijke (2008), Credibility improves topical blog post retrieval, in *Proceedings of ACL-08: HLT*, pp. 923–931, Association for Computational Linguistics (ACL).

Weimer, M., I. Gurevych, and M. Mühlhäuser (2007), Automatically assessing the post quality in online discussions on software, in *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pp. 125–128.

Wen, J., and H. Zhang (2002), Query clustering in the web context, *Information Retrieval and Clustering*, *20*(1), 59–81.

Wiebe, J., T. Wilson, and C. Cardie (2005), Annotating expressions of opinions and emotions in language, *Language Resources and Evaluation*, *39*(2), 165–210.

Wilson, T., P. Hoffmann, S. Somasundaran, J. Kessler, J. Wiebe, Y. Choi, C. Cardie, E. Riloff, and S. Patwardhan (2005), Opinionfinder: a system for subjectivity analysis, in *Proceedings of HLT/EMNLP on Interactive Demonstrations*, pp. 34–35, Association for Computational Linguistics.

Witten, I. H., G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning (1999), Kea: practical automatic keyphrase extraction, in *Proceedings of the 4th ACM Conference on Digital Libraries*, pp. 254–255, ACM.

Xi, W., J. Lind, and E. Brill (2004), Learning effective ranking functions for newsgroup search, in *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pp. 394–401, ACM, New York, NY, USA.

Xu, J., T. Liu, M. Lu, H. Li, and W. Ma (2008), Directly optimizing evaluation measures in learning to rank, in *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pp. 107–114, ACM, New York, NY, USA.

Yang, Y., and X. Liu (1999), A re-examination of text categorization methods, in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 42–49, ACM.

Yang, Y., and J. O. Pedersen (1997), A comparative study on feature selection in text categorization, in *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pp. 412–420, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Yeh, J., J. Lin, H. Ke, and W. Yang (2007), Learning to rank for information retrieval using genetic programming, in *Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*.

Yu, L., and H. Liu (2003), Feature selection for high-dimensional data: a fast correlation-based filter solution, in *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, vol. 20, pp. 856–863.

Zhang, Y., Y. Dang, and H. Chen (2011), Gender classification for web forums, *IEEE Transactions on Systems, Man, and Cybernetics*, *41*(4), 1–10.

Zheng, R., J. Li, H. Chen, and Z. Huang (2006), A framework for authorship identification of online messages: writing-style features and classification techniques, *Journal of the American Society for Information Science and Technology*, *57*(3), 378–393.