EFFECTIVE PATTERN DISCOVERY FOR TEXT DATA MINING

May, 2015

Submitted in partial fulfillment of the requirement for the Degree of

Bachelor of Technology

in

Information Technology

Under the Supervision of

Ms Ruchi Verma

by

Kunal Jhingrun(111446)

to

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

and INFORMATION TECHNOLOGY,

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT.

CERTIFICATE

This is to certify that the work titled "**EFFECTIVE PATTERN DISCOVERY FOR TEXT DATA MINING** submitted by **Kunal Jhingrun** in the partial fulfillment for the award of degree of Bachelor of Technology in Information Technology from Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor:

Name of Supervisor	:	Ms Ruchi Verma
Designation	:	Assistant Professor
Date	:	

ACKNOWLEDGEMENT

I would like to express my gratitude to all those who gave us the possibility to complete this project. I want to thank the Department of CSE & IT in JUIT for giving us the permission to commence this project in the first instance, to do the necessary research work.

I am deeply indebted to my project guide **Ms Ruchi Verma**, whose help, stimulating suggestions and encouragement helped me in all the time of research on this project. I feel motivated and encouraged every time I get his encouragement. For his coherent guidance throughout the tenure of the project, I feel fortunate to be taught by him, who gave me his unwavering support.

I am also grateful to **Mr.Amit Singh(CSE Project lab**) for his practical help and guidance.

Date: **15th May 2015** Kunal Jhingrun

Contents

Abbrevations and Symbols

IR – Information Retrieval

PTM – Pattern Taxonomy

IPE – Inner Pattern Evolution

RCV1 – Reuters Corpus Volume 1

TREC - Text Retrieval Conference

TFIDF - Term Frequency and Inverse Document Frequency

List of Figures

Title **Page No.**

List of Tables

Abstract

Due to the rapid growth of digital data made available in recent years, knowledge discovery and data mining have attracted a great deal of attention with an imminent need for turning such data into useful information and knowledge. Many applications, such as market analysis and business management, can benefit by the use of the information and knowledge extracted from a large amount of data.

Many data mining techniques have been proposed for mining useful patterns in text documents. However, how to effectively use and update discovered patterns is still an open research issue, especially in the domain of text mining. Since most existing text mining methods adopted term-based approaches, they all suffer from the problems of polysemy and synonymy. Over the years, people have often held the hypothesis that pattern (or phrase) based approaches should perform better than the term-based ones, but many experiments do not support this hypothesis. This paper presents an innovative and effective pattern discovery technique which includes the processes of pattern deploying and pattern evolving, to improve the effectiveness of using and updating discovered patterns for finding relevant and interesting information. Substantial experiments on RCV1 data collection and TREC topics demonstrate that the proposed solution achieves encouraging performance.

Since most existing text mining methods adopted term-based approaches, they all suffer from the problems of polysemy and synonymy. Over the years, people have often held the hypothesis that pattern (or phrase)-based approaches should perform better than the term-based ones, but many experiments do not support this hypothesis.
Disadvantages:
1. Phrases have inferior statistical properties to terms,
2. They have low frequency of occurrence, and
3. There are large numbers of redundant and noisy phrases among them

Here we present an effective pattern discovery technique, which first calculates discovered specificities of patterns and then evaluates term weights according to the distribution of terms in the discovered patterns rather than the distribution in documents for solving the misinterpretation problem. It also considers the influence of patterns from the negative training examples to find ambiguous (noisy) patterns and try to reduce their influence for the low-frequency problem. The process of updating ambiguous patterns can be referred as pattern evolution. The proposed approach can improve the accuracy of evaluating term weights because discovered patterns are more specific than whole documents.

Motivation

In the past decade, a significant number of data mining techniques have been presented in order to perform different knowledge tasks. These techniques include association rule mining, frequent item set mining, sequential pattern mining, maximum pattern mining and closed pattern mining. Most of them are proposed for the purpose of developing efficient mining algorithms to find particular patterns within a reasonable and acceptable time frame. With a large number of patterns generated by using data mining approaches, how to effectively use and update these patterns is still an open research issue.

In the field of text mining, pattern mining techniques can be used to find various text patterns, such as sequential patterns, frequent item sets, co-occurring terms and multiple grams, for building up a representation with these new types of features. Nevertheless, the challenging issue is how to effectively deal with the large amount of discovered patterns.[1]

Chapter – 1: An Overview

1.1 Introduction

Due to the rapid growth of digital data made available in recent years, knowledge discovery and data mining have attracted a great deal of attention with an imminent need for turning such data into useful information and knowledge. Many applications, such as market analysis and business management, can benefit by the use of the information and knowledge extracted from a large amount of data. Knowledge discovery can be viewed as the process of nontrivial extraction of information from large databases, information that is implicitly presented in the data, previously unknown and potentially useful for users. Data mining is therefore an essential step in the process of knowledge discovery in databases.

In the past decade, a significant number of data mining techniques have been presented in order to perform different knowledge tasks. These techniques include association rule mining, frequent item set mining, sequential pattern mining, maximum pattern mining and closed pattern mining. Most of them are proposed for the purpose of developing efficient mining algorithms to find particular patterns within a reasonable and acceptable time frame. With a large number of patterns generated by using data mining approaches, how to effectively use and update these patterns is still an open research issue. In this paper, we focus on the development of a knowledge discovery model to effectively use and update the discovered patterns and apply it to the field of text mining. Text mining is the discovery of interesting knowledge in text documents. It is a challenging issue to find accurate knowledge (or features) in text documents to help users to find what they want. In the beginning, Information Retrieval (IR) provided many term-based methods to solve this challenge, such as Rocchio and probabilistic models [4], rough set models , BM25 and SVM based filtering models. The advantages of term-based methods include efficient computational performance as well as mature theories for term weighting, which have emerged over the last couple of decades from the IR and machine learning communities. However, term-based methods suffer from the problems of polysemy and synonymy,

where polysemy means a word has multiple meanings, and synonymy is multiple words having the same meaning. The semantic meaning of many discovered terms is uncertain for answering what users want.

Although phrases are less ambiguous and more discriminative than individual terms, the likely reasons for the discouraging performance include:

(i) phrases have inferior statistical properties to terms,

(ii) they have low frequency of occurrence, and

(iii) there are large number of redundant and noisy phrases among them

In the presence of these setbacks, sequential patterns used in data mining community have turned out to be a promising alternative to phrases, because sequential patterns enjoy good statistical properties like terms. To overcome the disadvantages of phrase-based approaches, pattern mining based approaches (or pattern taxonomy models (PTM)) have been proposed, which adopted the concept of closed sequential patterns, and pruned non-closed patterns. These pattern mining based approaches have shown certain extent improvements on the effectiveness.

However, the paradox is that people think pattern-based approaches could be a significant alternative, but consequently less significant improvements are made for the effectiveness compared with term-based methods.[1]

1.2 Existing System

Since most existing text mining methods adopted term-based approaches, they all suffer from the problems of polysemy and synonymy. Over the years, people have often held the hypothesis that pattern (or phrase)-based approaches should perform better than the term-based ones, but many experiments do not support this hypothesis.

Although phrases are less ambiguous and more discriminative than individual terms, the likely reasons for the discouraging performance include:

(i) phrases have inferior statistical properties to terms,

(ii) they have low frequency of occurrence, and

(iii) there are large number of redundant and noisy phrases among them[1]

1.3 Proposed System:

Here we present an effective pattern discovery technique, which first calculates discovered specificities of patterns and then evaluates term weights according to the distribution of terms in the discovered patterns rather than the distribution in documents for solving the misinterpretation problem. It also considers the influence of patterns from the negative training examples to find ambiguous (noisy) patterns and try to reduce their influence for the low-frequency problem. The process of updating ambiguous patterns can be referred as pattern evolution. The proposed approach can improve the accuracy of evaluating term weights because discovered patterns are more specific than whole documents.[3]

Chapter 2 :Literature Survey

2.1. Summary of Papers

1 Paper Summary

Title of Paper	Effective Pattern Discovery for Text Mining
Authors	Ning Zhong, Yuefeng Li, and Sheng-Tang Wu
Year of Publication	2012
Publishing Details	IEEE Transactions on Knowledge and Data Engineering,Vol.24 No.1
Summary	Many data mining techniques have been proposed for mining useful patterns in text documents. However, how to effectively use and update discovered patterns is still an open research issue, especially in the domain of text mining. Since most existing text mining methods adopted term-based approaches, they all suffer from the problems of polysemy and synonymy. Over the years, people have often held the hypothesis that pattern (or phrase)-based approaches should perform better than the term-based ones, but many experiments do not support this hypothesis. This paper presents an innovative and effective pattern discovery technique which includes the processes of pattern deploying and pattern evolving, to improve the effectiveness of using and updating discovered patterns for finding relevant and interesting information.

	Substantial experiments on RCV1 data collection and TREC topics demonstrate results successfully.

Title of Paper	Effective Pattern Deploying Approach in Pattern Taxonomy Model for Text Mining
Authors	Rohini Y. Thombare , Shirish. S. Sane
Year of Publication	2013
Publishing Details	International Conference on Recent Trends in engineering & Technology - 2013
Summary	Text mining is a technique that helps user find useful information from a large amount of digital text document. Most existing text mining methods adopted term-based approaches, but they all suffer

	from the problem of polysemy and synonymy. The next phrase-based approach could not perform better than term based approach. Instead of using typical term-based method many data mining techniques have been proposed for mining useful patterns, however effective usage and updation of discovered patterns is still an open research issue. Pattern deploing and pattern evolving method has also been proposed in order to refine the patterns that helps in improving the effectiveness of pattern discovery. This paper presents an innovative pattern deploying technique based on pattern support to improve effectiveness of using and updating patterns. In existing method called PDM [1] it simply consider the number of sequential patterns containing the given term to compute term weight. The proposed method suggest a probabilistic method to estimate and compute term weight, in which we consider pattern support property which is already discovered in PTM model.

Chapter – 3: Data Mining

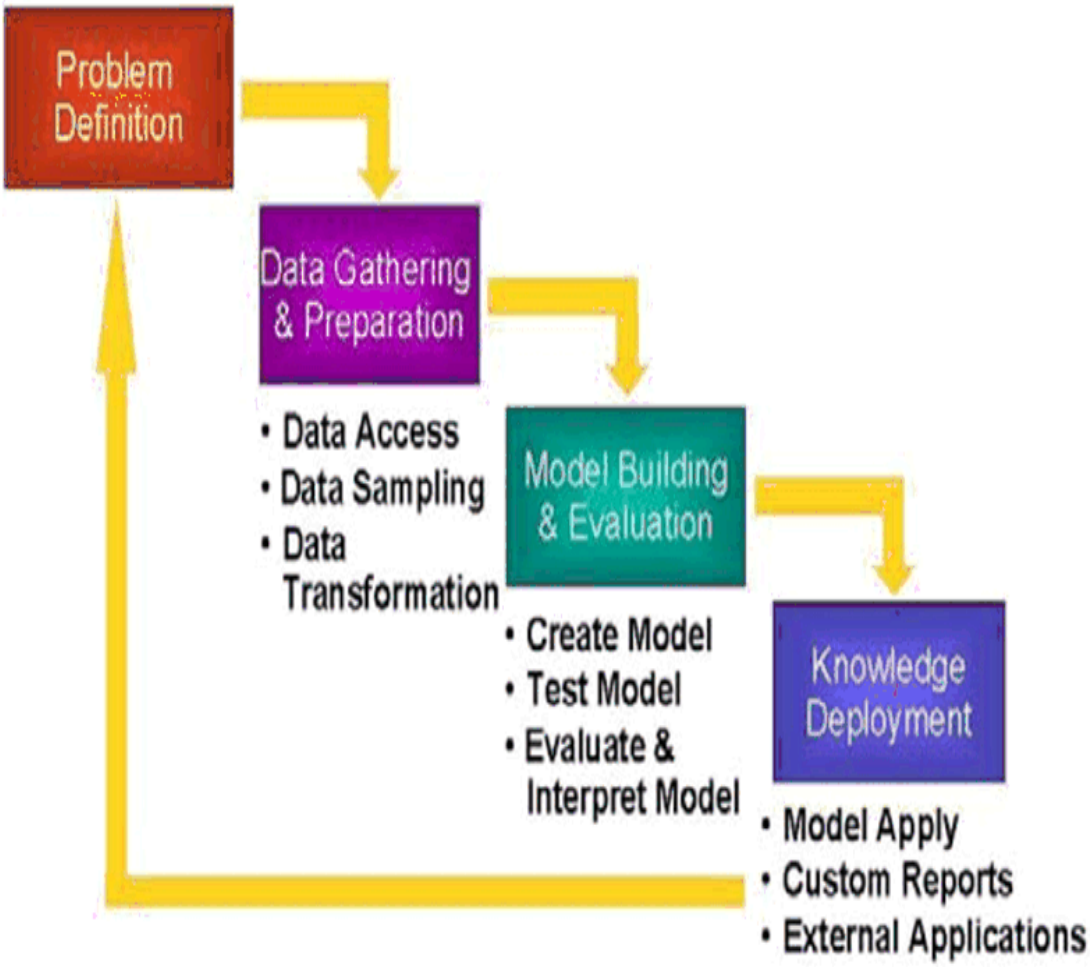

Figure 1

3.1 What is Data Mining?

Data mining (the analysis step of the "Knowledge Discovery in Databases" process, or KDD), an interdisciplinary subfield of computer science, is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. Aside from the raw analysis step, it involves database and data management aspects, data pre-processing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating.

The term is a misnomer, because the goal is the extraction of patterns and knowledge from large amount of data, not the extraction of data itself. It also is a buzzword, and is frequently also applied to any form of large-scale data or information processing (collection, extraction, warehousing, analysis, and statistics) as well as any application of computer decision support system, including artificial intelligence, machine learning, and business intelligence. The popular book "Data mining: Practical machine learning tools and techniques with Java" (which covers mostly machine learning material) was originally to be named just "Practical machine learning", and the term "data mining" was only added for marketing reasons. Often the more general terms "(large scale) data analysis", or "analytics" – or when referring to actual methods, artificial intelligence and machine learning– are more appropriate.

The actual data mining task is the automatic or semi-automatic analysis of large quantities of data to extract previously unknown interesting patterns such as groups of data records (cluster analysis), unusual records (anomaly detection) and dependencies (association rule mining). These patterns can then be seen as a kind of summary of the input data, and may be used in further analysis or, for example, in machine learning and predictive analytics.[4]

The key properties of data mining are:

- Automatic discovery of patterns
- Prediction of likely outcomes
- Creation of actionable information
- Focus on large data sets and databases

Data mining can answer questions that cannot be addressed through simple query and reporting techniques.

3.2What Can Data Mining Do and Not Do?

Data mining is a powerful tool that can help you find patterns and relationships within your data. But data mining does not work by itself. It does not eliminate the need to know your business, to understand your data, or to understand analytical methods. Data mining discovers hidden information in your data, but it cannot tell you the value of the information to your organization.

You might already be aware of important patterns as a result of working with your data over time. Data mining can confirm or qualify such empirical observations in addition to finding new patterns that may not be immediately discernible through simple observation.

It is important to remember that the predictive relationships discovered through data mining are not necessarily *causes* of an action or behavior. For example, data mining might determine that males with incomes between $50,000 and $65,000 who subscribe to certain magazines are likely to buy a given product. You can use this information to help you develop a marketing strategy. However, you should not assume that the population identified through data mining will buy the product *because* they belong to this population.

3.3 Tasks in Data Mining

Data mining involves six common classes of tasks:

- Anomaly detection (Outlier/change/deviation detection) – The identification of unusual data records, that might be interesting or data errors that require further investigation.

- Association rule learning (Dependency modeling) – Searches for relationships between variables. For example a supermarket might gather data on customer purchasing habits. Using association rule learning, the supermarket can determine which products are frequently bought together and use this information for marketing purposes. This is sometimes referred to as market basket analysis.

- Clustering – is the task of discovering groups and structures in the data that are in some way or another "similar", without using known structures in the data.

- Classification – is the task of generalizing known structure to apply to new data. For example, an e-mail program might attempt to classify an e-mail as "legitimate" or as "spam".

- Regression – attempts to find a function which models the data with the least error.

- Summarization – providing a more compact representation of the data set, including visualization and report generation.

3.4 How does data mining work?

While large-scale information technology has been evolving separate transaction and analytical systems, data mining provides the link between the two. Data mining software analyzes relationships and patterns in stored transaction data based on open-ended user queries. Several types of analytical software are available: statistical, machine learning, and neural networks. Generally, any of four types of relationships are sought:

- **Classes**: Stored data is used to locate data in predetermined groups. For example, a restaurant chain could mine customer purchase data to determine when customers visit and what they typically order. This information could be used to increase traffic by having daily specials.

- **Clusters**: Data items are grouped according to logical relationships or consumer preferences. For example, data can be mined to identify market segments or consumer affinities.

- **Associations**: Data can be mined to identify associations. The beer-diaper example is an example of associative mining.

- **Sequential patterns**: Data is mined to anticipate behavior patterns and trends. For example, an outdoor equipment retailer could predict the likelihood of a backpack being purchased based on a consumer's purchase of sleeping bags and hiking shoes.

Data mining consists of five major elements:

- Extract, transform, and load transaction data onto the data warehouse system.
- Store and manage the data in a multidimensional database system.
- Provide data access to business analysts and information technology professionals.
- Analyze the data by application software.
- Present the data in a useful format, such as a graph or table.

3.5 Pattern Mining

"Pattern mining" is a data mining method that involves finding existing patterns in data. In this context patterns often means association rules. The original motivation for searching association rules came from the desire to analyze supermarket transaction data, that is, to examine customer behavior in terms of the purchased products. For example, an association rules "beer ⇒ potato chips (80%)" states that four out of five customers that bought beer also bought potato chips.

In the context of pattern mining as a tool to identify terrorist activity, the National Research Council provides the following definition: "Pattern-based data mining looks for patterns (including anomalous data patterns) that might be associated with terrorist activity — these patterns might be regarded as small signals in a large ocean of noise." Pattern Mining includes new areas such a Music Information Retrieval (MIR) where patterns seen both in the temporal and non temporal domains are imported to classical knowledge discovery search methods.[4]

Pattern mining is one of the most important topics in data mining. The core idea is to extract relevant "nuggets" of knowledge describing parts of a database. However, many

traditional (frequent) pattern mining algorithms find patterns in numbers too large to be of practical value: so many "nuggets" of knowledge are found that they do not combine into a better global understanding of the data. In fact, often the number of discovered patterns is larger than the size of the original database.

Chapter – 4: Text Data Mining

4.1 What is Text Mining?

Text mining, also referred to as text data mining, roughly equivalent to text analytics, refers to the process of deriving high-quality information from text. High-quality information is typically derived through the devising of patterns and trends through means such as statistical pattern learning. Text mining usually involves the process of structuring the input text (usually parsing, along with the addition of some derived linguistic features and the removal of others, and subsequent insertion into a database), deriving patterns within the structured data, and finally evaluation and interpretation of the output. 'High quality' in text mining usually refers to some combination of relevance, novelty, and interestingness. Typical text mining tasks include text categorization, text clustering, concept/entity extraction, production of granular taxonomies, sentiment analysis, document summarization, and entity relation modeling (*i.e.*, learning relations between named entities).

Text analysis involves information retrieval, lexical analysis to study word frequency distributions, pattern recognition, tagging/annotation, information extraction, data mining techniques including link and association analysis, visualization, and predictive analytics. The overarching goal is, essentially, to turn text into data for analysis, via application of natural language processing (NLP) and analytical methods.

A typical application is to scan a set of documents written in a natural language and either model the document set for predictive classification purposes or populate a database or search index with the information extracted.[7]

Text mining can help an organization derive potentially valuable business insights from text-based content such as word documents, email and postings on social media streams like Facebook, Twitter and LinkedIn. Mining unstructured data with natural language

processing (NLP), statistical modeling and machine learning techniques can be challenging, however, because natural language text is often inconsistent. It contains ambiguities caused by inconsistent syntax and semantics, including slang, language specific to vertical industries and age groups, double entendres and sarcasm.

Text analytics software can help by transposing words and phrases in unstructured data into numerical values which can then be linked with structured data in a database and analyzed with traditional data mining techniques. With an iterative approach, an organization can successfully use text analytics to gain insight into content-specific values such as sentiment, emotion, intensity and relevance. Because text analytics technology is still considered to be an emerging technology, however, results and depth of analysis can vary wildly from vendor to vendor.

4.2 Text Mining Process

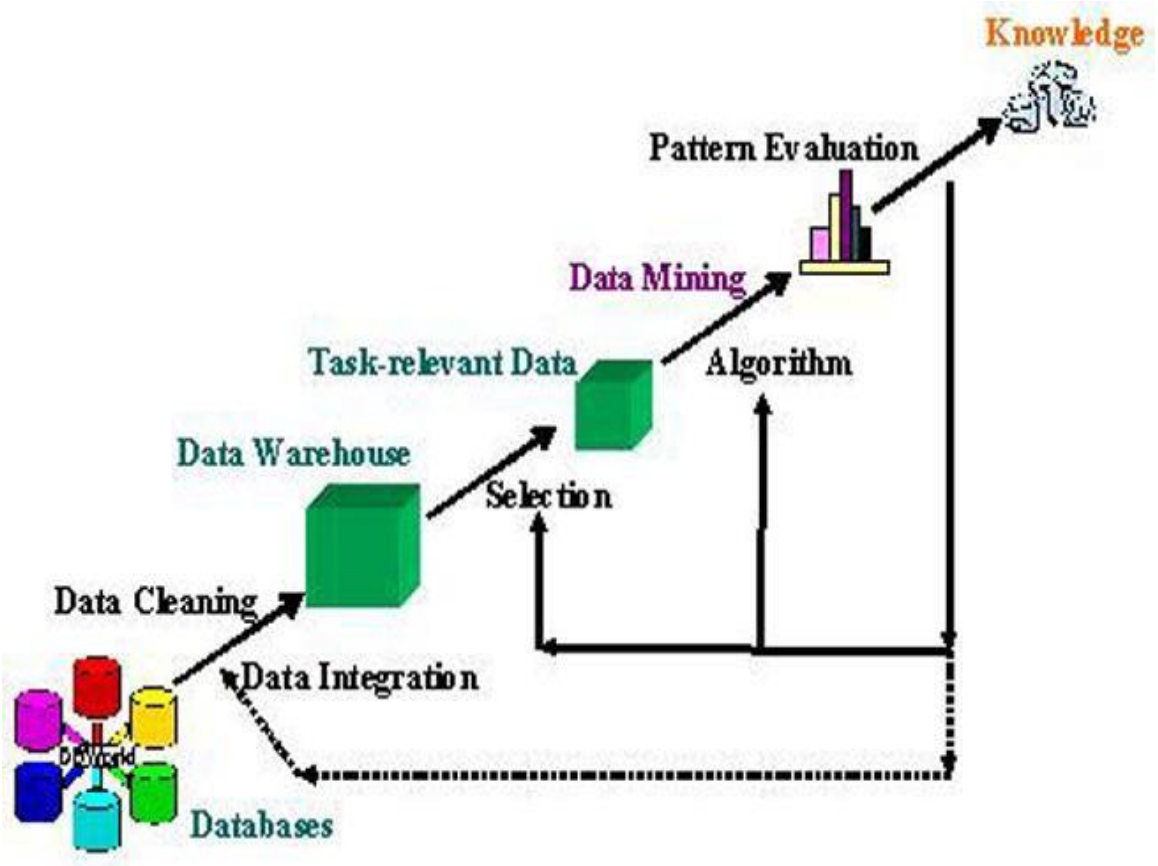

Figure 2

4.3 Application of Text Mining

The technology is now broadly applied for a wide variety of government, research, and business needs. Applications can be sorted into a number of categories by analysis type or by business function. Using this approach to classifying solutions, application categories include:

- Enterprise Business Intelligence/Data Mining, Competitive Intelligence
- E-Discovery, Records Management
- National Security/Intelligence
- Scientific discovery, especially Life Sciences
- Sentiment Analysis Tools, Listening Platforms
- Natural Language/Semantic Toolkit or Service
- Publishing
- Automated ad placement
- Search/Information Access
- Social media monitoring

Chapter – 5: Pattern Taxonomy Model

Two main stages are considered in PTM. The first stage is how to extract useful phrases from text documents, and the second stage is then how to use these discovered patterns to improve the effectiveness of a knowledge discovery system.

In PTM, we split a text document into a set of paragraphs and treat each paragraph as an individual transaction, which consists of a set of words (terms). At the subsequent phase, we apply the data mining method to find frequent patterns from these transactions and generate pattern taxonomies. Pattern taxonomy is a tree like structure that defines the relationship between patterns extracted from text collection. An example of pattern taxonomy is shown in figure 1. The arrow in figure indicates the sub-sequence relation between patterns. For example, pattern <A, B> is a sub-sequence of pattern <A, B, C>, and pattern <B> is a sub-sequence of pattern <B, C>. the root of the tree in the bottom represents longest patterns (i.e. maximum sequential patterns). During the next pruning phase, non-meaningful and redundant patterns are eliminated.[3]

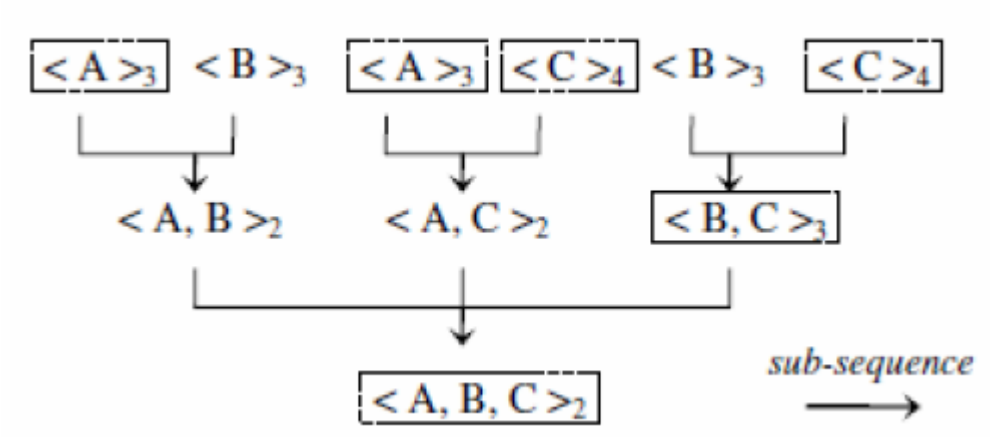

Figure 3

As can be seen in the Figure 2, pattern <A, B> is closed pattern of <A, B, C>. That means they always appear in the same paragraph. Therefore, the shorter one i.e. pattern <A, B> is negligible and is considered as a meaningless pattern.

5.1 Frequent and Closed pattern

A sequential pattern P is called frequent sequential pattern if support (P) is greater than or equal to a minimum support (min_sup) ξ.

For example, let min_sup $\xi = 0.75$ for the document shown in Table 1; we can obtain four frequent sequential patterns: <B,C>, <A>, <B>, and <C> since their relative supports are not less than ξ.

The purpose of using min_sup in our project is to reduce the number of patterns discovered in a large document. Otherwise these patterns with lower relative support will increase the burden of the training.[3]

A set of paragraphs

Paragraph	Terms
dp_1	t_1 t_2
dp_2	t_3 t_4 t_6
dp_3	t_3 t_4 t_5 t_6
dp_4	t_3 t_4 t_5 t_6
dp_5	t_1 t_2 t_6 t_7
dp_6	t_1 t_2 t_6 t_7

Table 1

Above table lists a set of paragraphs for a given document d, where PS(d) = {dp1, dp2,… , dp6}, and duplicate terms were removed. Let min sup = 50%, we can obtain ten frequent patterns in Table 1 using the above definitions.

Frequent patterns and covering sets

2 Frequent patterns and covering sets

Frequent Pattern	Covering Set
{t_3, t_4, t_6}	{dp_2, dp_3, dp_4}
{ t_3, t_4}	{ dp_2, dp_3, dp_4}
{ t_3, t_6}	{ dp_2, dp_3, dp_4}
{ t_4, t_6}	{ dp_2, dp_3, dp_4}
{ t_3 }	{ dp_2, dp_3, dp_4}
{ t_4}	{ dp_2, dp_3, dp_4}
{t_1, t_2}	{ dp_1, dp_5, dp_6}
{ t_1 }	{ dp_1, dp_5, dp_6}
{ t_2}	{ dp_1, dp_5, dp_6}
{ t_6}	{ dp_2, dp_3, dp_4, dp_5, dp_6}

Table2

Not all frequent patterns in Table are useful. For example, pattern ft3; t4g always occurs with term t6 in paragraphs, i.e., the shorter pattern, {t3; t4}, is always a part of the larger pattern, ft3; t4; t6g, in all of the paragraphs. Hence, we believe that the shorter one, {t3; t4}, is a noise pattern and expect to keep the larger pattern, {t3; t4; t6}, only.

5.2 Closed Sequential Pattern

A frequent closed sequential pattern is a frequent sequential pattern such that it is not included in another sequential pattern having exactly the same support.

Let's consider 4 sequences:

```
a b c d e
a b d
b e a c
b c d e
```

Let's say that minsup = 2.

b c is a frequent sequential patterns because it appears in two sequences (it has a support of 2). b c is not a closed sequential patterns because it is contained in a larger sequential pattern b c d having the same support.

b c d has a support of 2. It is also not a closed sequential pattern because it is contained in a larger sequential pattern b c d e having the same support. b c d e is a closed sequential pattern because there it is not included in any other sequential pattern having the same support.

So, a pattern is closed if none of its immediate supersets has the same support as the pattern.[1]

Chapter – 6: Pattern Deploying Method

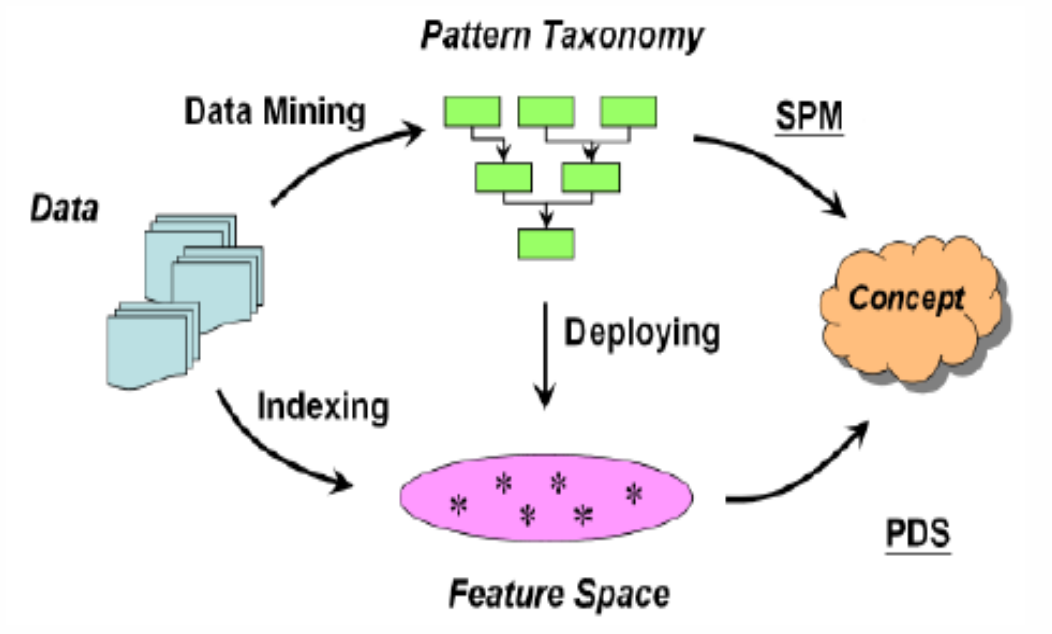

Figure 3. Flowchart of Pattern deploying method in PTM.

Figure 4

The pattern taxonomy methodology

In order to use semantic information in the pattern taxonomy to improve the performance of closed patterns in text mining, we need to interpret discovered patterns in order to accurately evaluate term weights (supports). The rational behind this motivation is that discovered patterns include more semantic meaning than the terms that are selected based on a term-based technique (e.g., tf*idf). In term-based approaches, the evaluations of term weights (supports) are based on the distribution of terms in documents. The evaluation of term weights (supports) is different to the normal term-based approaches. As suggested in , in deploying method, terms are weighted according to their appearances in discovered closed patterns. It simply deploy patterns through the use of a pattern composition operator.[1]

6.1 D-Pattern Mining Algorithm

To improve the efficiency of the pattern taxonomy mining, an algorithm, SPMining, was proposed in to find all closed sequential patterns, which used the well-known Apriori property in order to reduce the searching space. For every positive document, the SPMining algorithm is first called giving rise to a set of closed sequential patterns SP. The main focus of this paper is the deploying process, which consists of the d-pattern discovery and term support evaluation. All discovered patterns in a positive document are composed into a d-pattern giving rise to a set of d-patterns DP.

Thereafter term supports are calculated based on the normal forms for all terms in d-patterns. In Sequential Pattern Mining (SPM) algorithm SPMining we apply the pruning scheme for the purpose of eliminating non-closed patterns during the process of sequential patterns discovery. The algorithm repeats itself recursively until there is no more pattern discovered. As a result, the output of algorithm SPMining is a set of closed sequential patterns with relative supports greater than or equal to a specified minimum support. The experimental results showed that Pattern Taxonomy Model (PTM) is a feasible way to apply data mining techniques to the text mining area.

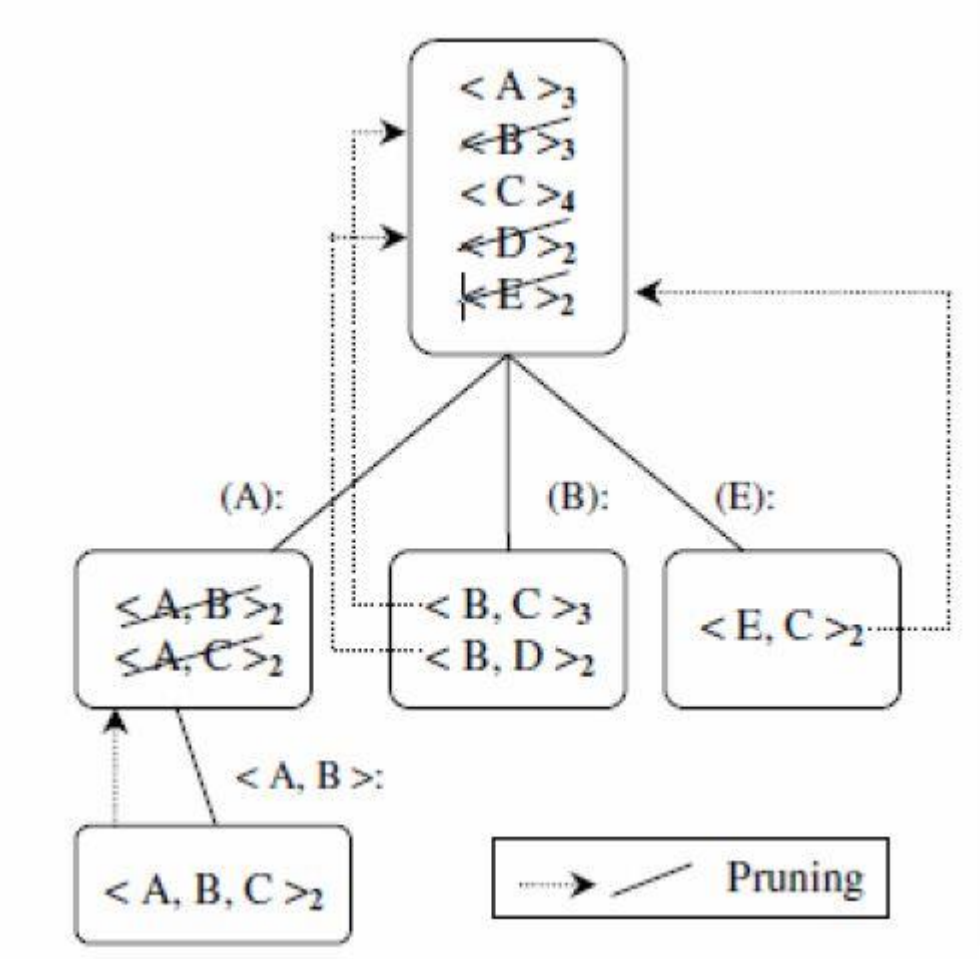

Figure 5

Pruning of patterns

6.2 PTM Algorithm

To simplify this process, we firstly review the composition operation Θ
The result of the composition is still a set
of term-number pairs.
For example,
{(t1, 1), (t2, 2); (t3, 3)} Θ {(t2, 4)} = {(t1, 1), (t2, 6), (t3, 3)}
or
{(t1, 2%), (t2, 5%), (t3, 9%)} Θ {(t1, 1%), (t2, 3%)}
= {(t1, 3%), (t2, 8%), (t3, 9%)}

input : positive documents D+; minimum support,
min sup.
output: d-patterns DP, and supports of terms.
DP = Ø;
foreach document d € D+ do
let PS(d) be the set of paragraphs in d;
SP = SPMining(PS(d), min sup);
db= Ø ;
foreach pattern pi € SP do
p = {(t; 1)|t € pi};
db= db Θ p;
end
DP = DP U {db};
end
T = {t|(t, f) € p, p € D};
foreach term t € T do
support(t) = 0;

```
    end
    foreach d-pattern p 2 DP do
            foreach (t,w) € β(p) do
                    support(t) = support(t) + w;
                end
end
```

Chapter – 7: Apriori Algorithm

Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation). Other algorithms are designed for finding association rules in data having no transactions (Winepi and Minepi), or having no timestamps (DNA sequencing). Each transaction is seen as a set of items (an itemset). Given a threshold C, the Apriori algorithm identifies the item sets which are subsets of at least C transactions in the database.

Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as candidate generation), and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found.

Apriori uses breadth-first search and a Hash tree structure to count candidate item sets efficiently. It generates candidate item sets of length k from item sets of length $k-1$. Then it prunes the candidates which have an infrequent sub pattern. According to the downward closure lemma, the candidate set contains all frequent k-length item sets. After that, it scans the transaction database to determine frequent item sets among the candidates.

For example, the rule {onion,potatoes}=>{burger} found in the sales data of a supermarket would indicate that if a customer buys onions and potatoes together, he or she is likely to also buy burger. Such information can be used as the basis for decisions about marketing activities such as, e.g., promotional pricing or product placements.

In addition to the above example from market basket analysis association rules are employed today in many application areas including Web usage mining, intrusion detection and bioinformatics.

In computer science and data mining, Apriori is a classic algorithm for learning association rules. Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation).

To illustrate the concepts, we use a small example from the supermarket domain. The set of items is I = {milk,bread,butter,beer} and a small database containing the items (1 codes presence and 0 absence of an item in a transaction) is shown in the table below. An example rule for the supermarket could be {milk,bread}=>{butter} meaning that if milk and bread is bought, customers also buy butter.

Note: this example is extremely small. In practical applications, a rule needs a support of several hundred transactions before it can be considered statistically significant, and datasets often contain thousands or millions of transactions.

Transaction ID	milk	Bread	butter	beer
1	1	1	0	0
2	0	1	1	0
3	0	0	0	1
4	1	1	1	0
5	0	1	0	0

Figure 6

7.1 Support

The support **supp(*X*)** of an itemset *X* is defined as the proportion of transactions in the data set which contain the itemset.

supp(X)= no. of transactions which contain the itemset X / total no. of transactions

In the example database, the itemset {milk,bread,butter} has a support of 4 /15 = 0.26 since it occurs in 26% of all transactions. To be even more explicit we can point out that 4 is the number of transactions from the database which contain the itemset {milk,bread,butter} while 15 represents the total number of transactions.

7.2 Confidence

The confidence of a rule is defined:

Confidence ($X \rightarrow Y$) = support($X \cup Y$)/support(X)

For the rule {milk,bread}=>{butter} we have the following confidence:

supp({milk,bread,butter}) / supp({milk,bread}) = 0.26 / 0.4 = 0.65

This means that for 65% of the transactions containing milk and bread the rule is correct. Confidence can be interpreted as an estimate of the probability P(Y | X), the probability of finding the RHS of the rule in transactions under the condition that these transactions also contain the LHS.

Chapter – 8: Inner Pattern Evolution

In the previous section, PTM model has been significantly improved after the adoption of pattern deploying method PDS, in order to solve low frequency problem. However, information from the negative examples has not been exploited during the concept learning there is no doubt that negative documents contains much useful information to identify ambiguous patterns in the concept. In this section, we discuss how to reshuffle support of terms within normal forms of discovered patterns based on negative documents in the training set. The technique will be useful to reduce the side effects of noisy patterns because of the low- frequency problem. This technique is called inner pattern evolution, A set of interesting negative documents, labeled as relevant by the system, is first detected. Two types of offenders can be discovered from these interesting negative documents: total conflict and partial conflict.

The basic idea of updating patterns is explanned as follow: total conflict offenders are removed from discovered patterns. For partial conflict offenders, their term supports are reshuffled in order to reduce the effects of noise documents. The main process of inner pattern evolution is implemented by the algorithm IPEvolving. The inputs of this algorithm are a set of discovered patterns DP, a training set D = D+ U D-. The output is a composed of discovered pattern. The second step in IPEvolving is used to estimate the threshold for finding the noise negative documents. Step 3 to 10 revise term supports by using all noise negative documents. Step 4 is to find noise documents and the corresponding offenders. Step 5 gets normal forms of discovered patterns NDP. Step 6 calls algorithm shuffling to update NDP according to noise documents. The task of algorithm shuffling is to tune the support distribution of terms within a discovered pattern. Steps 7 to 9 compose updated normal forms together. The advantage od IPE is that not all sequential patterns are necessary to be involved during the evolving process. Only those that are also found in the negative documents need to be re-evaluated. As the result, the efficiency of the system can be improved.[1]

8.1 Flowchart of Pattern Evolving Approach

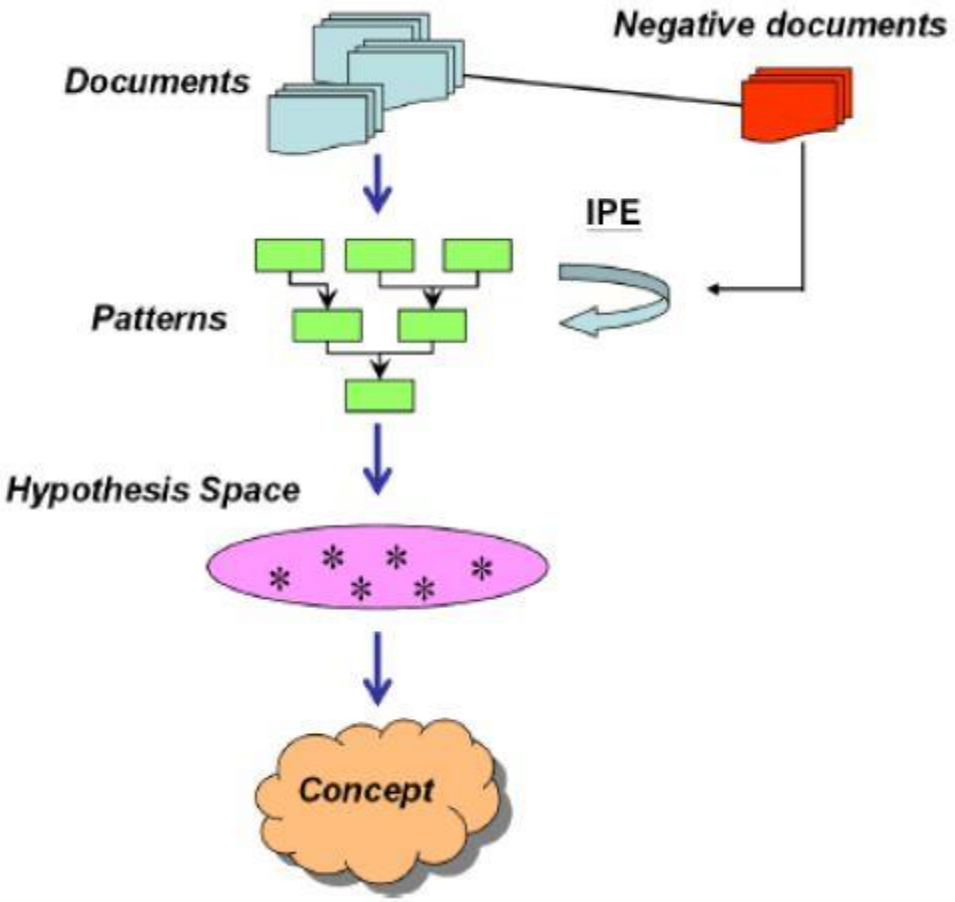

Figure 7

8.2 IPEvolving Algorithm

```
   input : a training set D = D+ ∪ D−; a set of
           d-patterns DP; and an experimental coefficient
           μ.
   output: a set of term-support pairs np.
 1 np ← ∅;
 2 threshold = Threshold(DP);// see Eq. (5)
 3 foreach noise negative document nd ∈ D− do
 4 |   if weight(nd) ≥ threshold then
   |   Δ(nd) = {p ∈ DP|termset(p) ∩ nd ≠ ∅};
 5 |   NDP = {β(p)|p ∈ DP};
 6 |   Shuffling(nd, Δ(nd), NDP, μ. NDP); //call Alg. 3
 7 |   foreach p ∈ NDP do
 8 |   |   np ← np ⊕ p;
 9 |   end
10 end
```

IPEvolving(D+, D-, DP, μ)

The main process of inner pattern evolution is implemented by the algorithm IPEvolving. The inputs of this algorithm are a set of d patterns DP, a training set D = D+ U D-. The output is a composed d-pattern. Step (2) in IPEvolving is used to estimate the threshold for finding the noise negative documents. Step 3 to Step 10 revise term supports by using all noise negative documents. Step (4) is to find noise documents and the corresponding offenders. Step (5) gets normal forms of d-patterns NDP. Step (6) calls algorithm shuffling to update NDP according to noise documents. Steps (7) to (9) compose updated normal forms together.

8.3 Shuffling Algorithm

The task of algorithm Shuffling is to tune the support distribution of terms within a d-pattern. A different strategy is dedicated in this algorithm for each type of offender. As stated in step (2) in the algorithm Shuffling, complete conflict offenders (d-patterns) are removed since all elements within the d-patterns are held by the negative documents indicating that they can be discarded for preventing interference from these possible "noises".

input : a noise document nd, its offenders $\Delta(nd)$,
normal forms of d-patterns NDP, and an
experimental coefficient μ.
output: updated normal forms of d-patterns NDP.

```
1  foreach d-pattern p in Δ(nd) do
2      if termset(p) ⊆ nd then NDP = NDP − {β(p)};
       //remove complete conflict offenders
3      else //partial conflict offender
4          offering =
           (1 − 1/μ) × Σ_{t∈(termset(p)∩nd)} support(t);
           base = Σ_{t∈(termset(p)−nd)} support(t);
5          foreach term t in termset(p) do
6              if t ∈ nd then
7              support(t) = (1/μ) × support(t); //shrink
8              else //grow supports
9                  support(t) =
                   support(t) × (1 + offering ÷ base);
10
11         end
12
13 end
```

Chapter – 9: Conclusion

In the last decade, many data mining techniques have been proposed for fulfilling various knowledge discovery tasks. These techniques include association rule mining, frequent item set mining, sequential pattern mining, maximum pattern mining and closed pattern mining. However using these discovered patterns in the field of text mining is difficult and ineffective. The reason is that a useful long pattern with high specificity lacks in support. We argue that not all frequent short patterns are useful. Hence, inadequate use of patterns derived from data mining techniques leads to the ineffective performance. In this paper, an effective pattern taxonomy model has been proposed to overcome the problem of deploying discovered patterns into a hypothesis space. In addition, pattern updating schemes are also investigated.

This paper presents the research on the concept of developing an effective knowledge discovery model (PTM) based on pattern taxonomies. PTM is implemented by three main steps: 1) discovering useful patterns by integrating sequential closed pattern mining algorithm and pruning scheme; 2) using discovered pattern deploying; 3) adjusting user profiles by applying pattern evolution. Various mechanisms in each step are proposed and evaluated for fulfilling the PTM model. A pattern deploying method based on support (PDS) is developed to deal with the discovered patterns in proper ways and provide suitable solutions for using these patterns. Knowledge based system many useful features such as support and confidence of a pattern, relationship between patterns, distribution of pattern taxonomies, and the dimension of these taxonomies are provided. In PTM system, some features such as the relationship among patterns and support of patterns have been studied. The rest of the features will be used in further research work. Most of the data mining algorithms are computationally expensive such as PTM, especially during the phase of Pattern Deploying. One possible solution to improve the efficiency of pattern taxonomy-based model is to reduce the dimensionality of the feature

space in the knowledge base. One alternative solution is to apply length-decreasing support constraints to frequent pattern mining.

So far I have calculated the d-patterns from a positive document, in future work I will remove the noisy patterns from the d-patterns obtained in order to improve its use and compare the results obtained with various algorithms.

Chapter – 10: Implementation

10.1 Main Modules

10.1.1 PATTERN TAXONOMY MODEL

We assume that all documents are split into paragraphs. So a given document d yields a set of paragraphs. Let D be a training set of documents, which consists of a set of positive documents and a set of negative documents.

10.1.2 PATTERN DEPLOYING METHOD

In order to use the semantic information in the pattern taxonomy to improve the performance of closed patterns in text mining, we need to interpret discovered patterns by summarizing them as d-patterns (see the definition below) in order to accurately evaluate term weights (supports). The rational behind this motivation is that d-patterns include more semantic meaning than terms that are selected based on a term-based technique (e.g., tf*idf).

10.2 Flow Chart

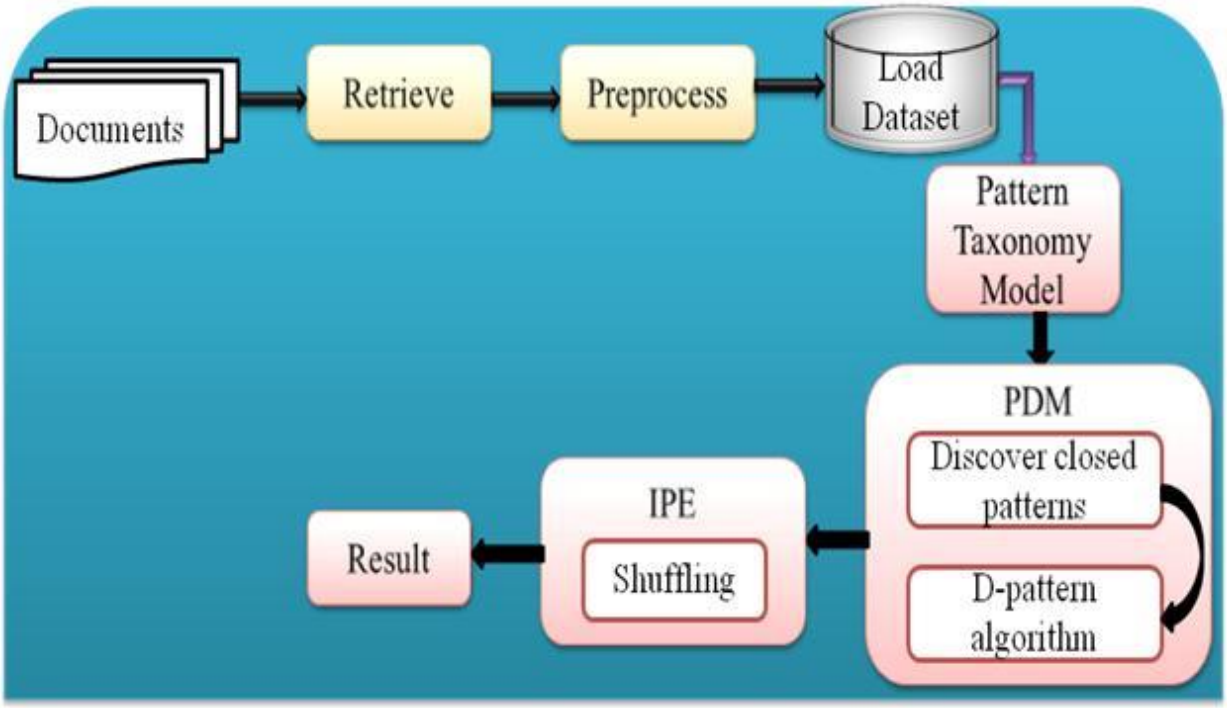

Figure 8

10.3 Use Case Diagram

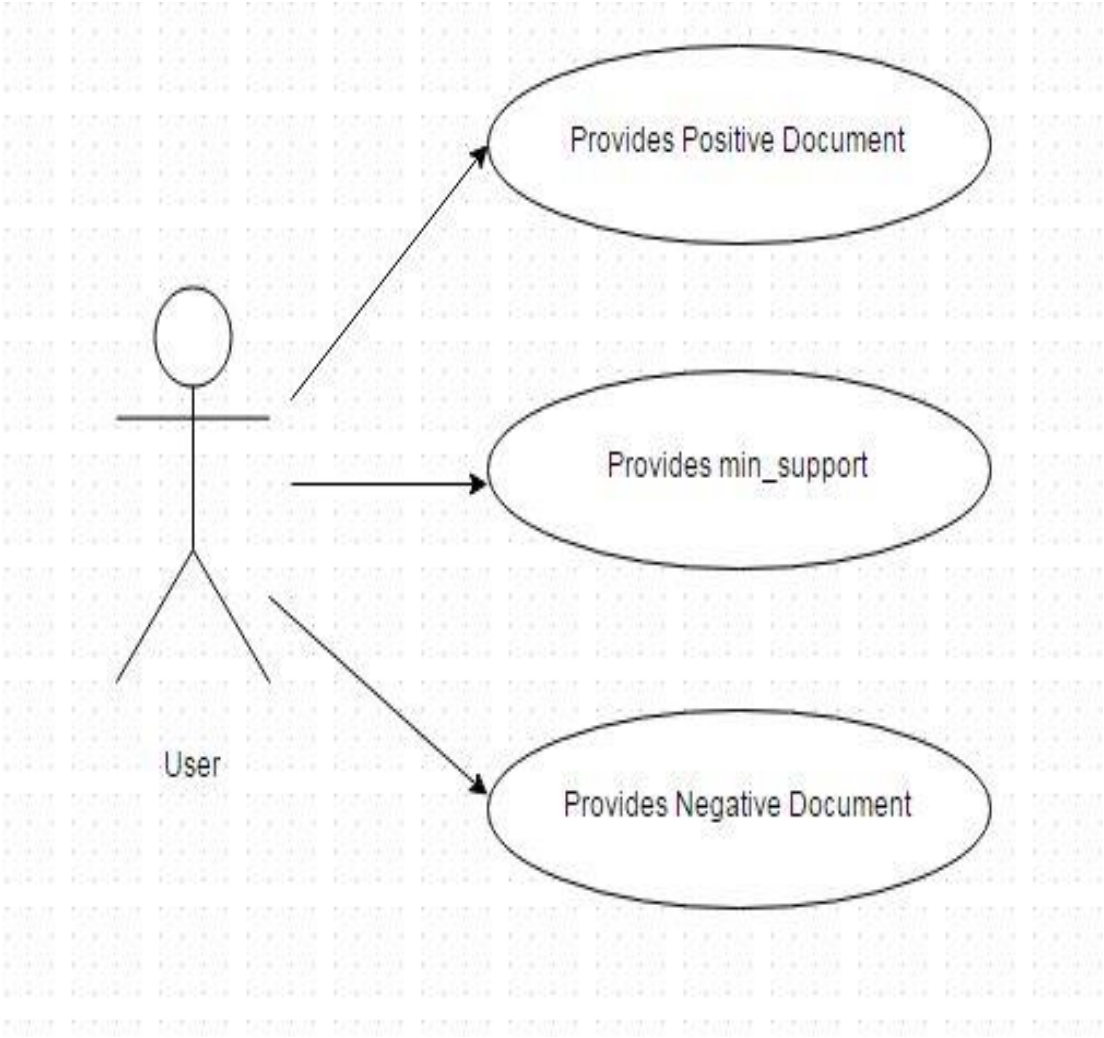

Figure 9

10.4 Screen Shots

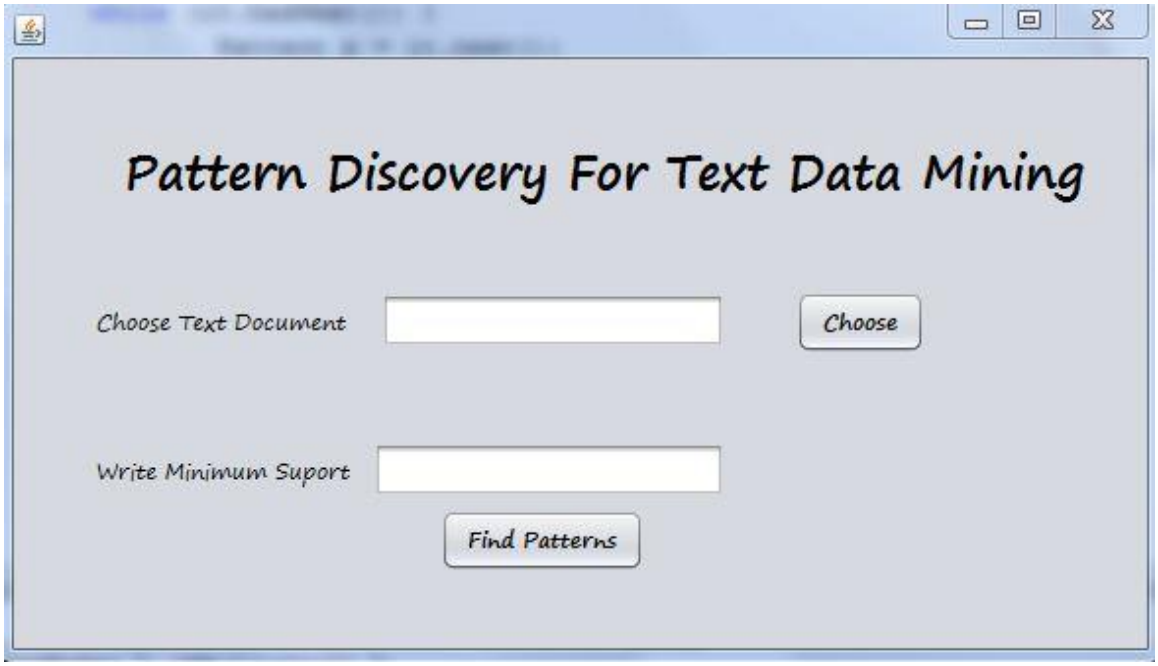

Figure 10

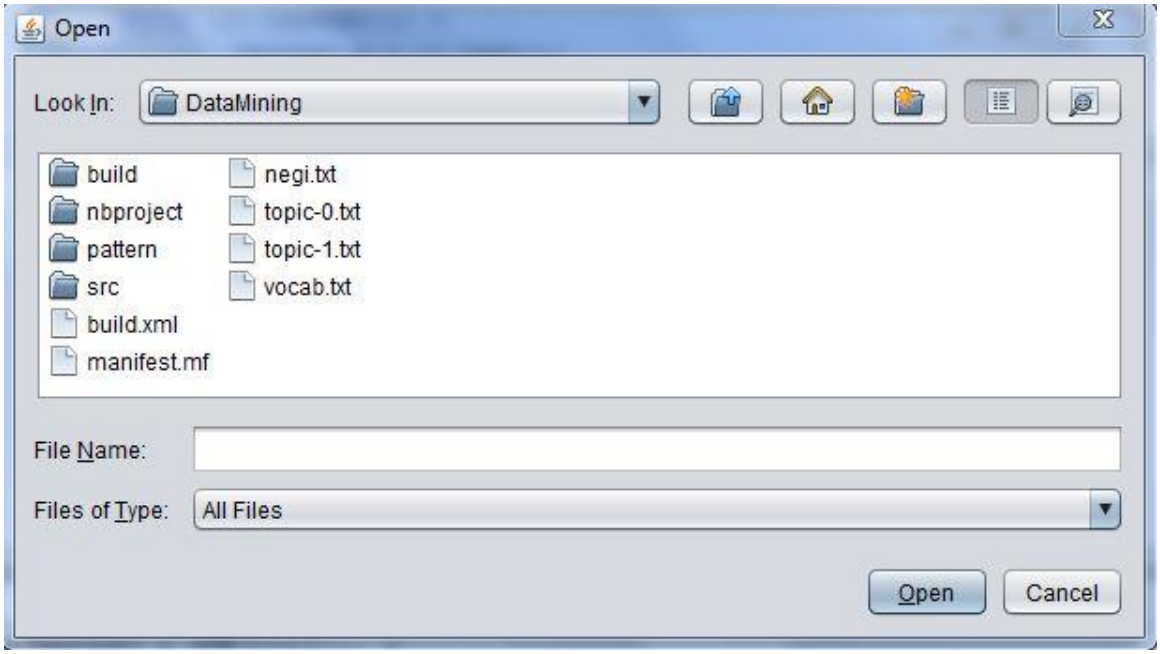

Figure 11

```
d-pattern: 107 algorithm efficient
d-pattern: 73 based data
d-pattern: 43 based clustering
d-pattern: 54 based mining
d-pattern: 54 based algorithm
d-pattern: 233 rule association
d-pattern: 159 mining association
```

Figure 12

Chapter 11: Testing

A primary purpose of testing is to detect software failures so that defects may be discovered and corrected. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions. The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions as well as examining the aspects of code: does it do what it is supposed to do and do what it needs to do. In the current culture of software development, a testing organization may be separate from the development team. There are various roles for testing team members. Information derived from software testing may be used to correct the process by which software is developed.

11.1 Types of Testing

11.1.1 Functional Testing

Functional testing is a quality assurance (QA) process and a type of black box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white-box testing).[2] Functional testing usually describes what the system does.

Functional testing does not imply that you are testing a function (method) of your module or class. Functional testing tests a slice of functionality of the whole system.

Functional testing differs from system testing in that functional testing "verifies a program by checking it against design document(s) or specification(s)", while system testing "validate[s] a program by checking it against the published user or system requirements"

11.1.2 Structural Testing

White-box testing also known as clear box testing, glass box testing, transparent box testing, and structural testing is a method of testing the application at the level of the source code. These test cases are derived through the use of the design techniques mentioned above:control flow testing, data flow testing, branch testing, path testing, statement coverage and decision coverage as well as modified condition/decision coverage. White-box testing is the use of these techniques as guidelines to create an error free environment by examining any fragile code. These White-box testing techniques are the building blocks of white-box testing, whose essence is the careful testing of the application at the source code level to prevent any hidden errors later on.[1] These different techniques exercise every visible path of the source code to minimize errors and create an error-free environment. The whole point of white-box testing is the ability to know which line of the code is being executed and being able to identify what the correct output should be.

While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

11.1.3 Unit Testing

Unit testing, also known as component testing, refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to ensure that the building blocks of the software work independently from each other.

Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional QA focuses, it augments it. Unit testing aims to eliminate construction errors before code is promoted to QA; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and QA process.

11.2 Sample Output

```
d-pattern: 74 learning bayesian
d-pattern: 97 learning network
d-pattern: 56 learning approach
d-pattern: 80 learning supervised
d-pattern: 147 learning using
d-pattern: 61 learning multi
d-pattern: 52 model mixture
d-pattern: 54 classification feature
d-pattern: 94 classification using
d-pattern: 64 model markov
d-pattern: 101 network neural
d-pattern: 58 model probabilistic
d-pattern: 98 decision tree
d-pattern: 146 feature selection
d-pattern: 55 using feature
d-pattern: 50 clustering hierarchical
d-pattern: 121 neighbor nearest
d-pattern: 58 using clustering
d-pattern: 120 using model
d-pattern: 107 network bayesian
d-pattern: 76 using network
d-pattern: 76 using recognition
d-pattern: 115 support machine vector
d-pattern: 51 learning semi supervised
Total: 171 patterns.
Mining topic-1 Finished.


***********Frequent Pattern Mining Finished***********
```

Figure 13

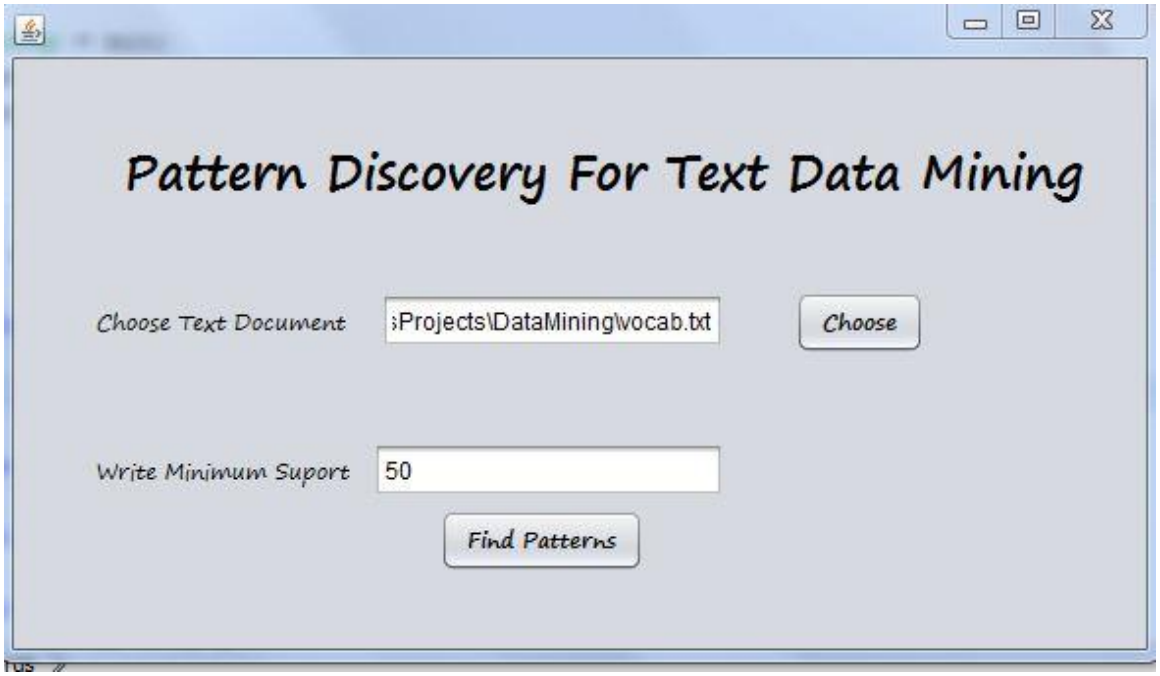

Figure 14

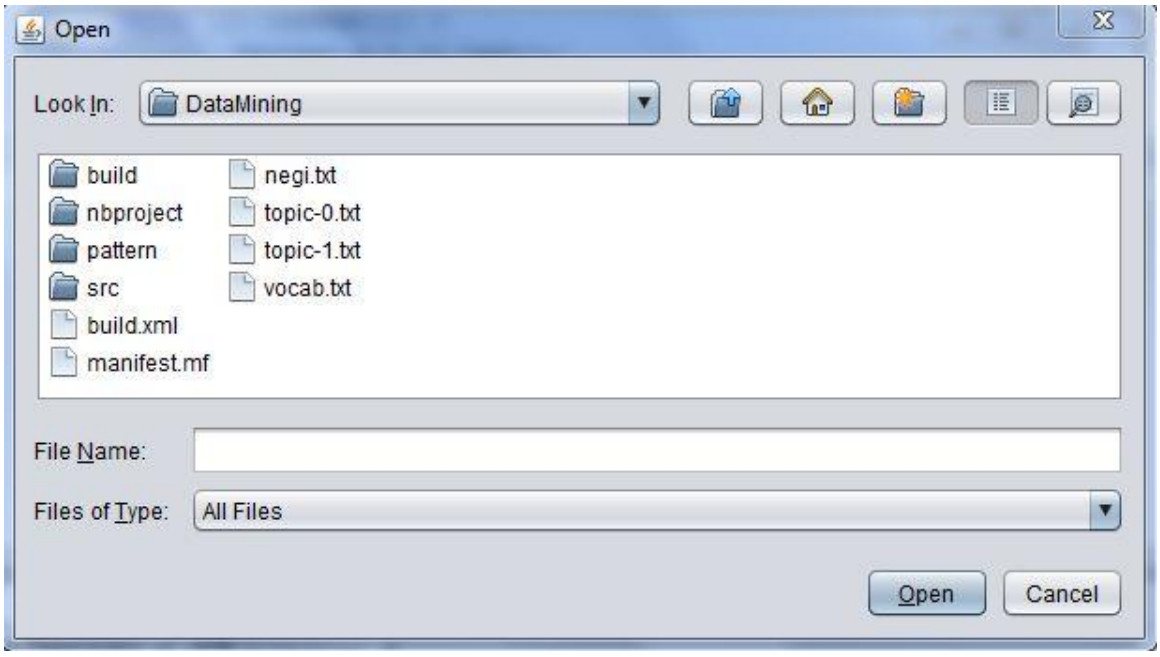

Figure 15

Chapter 12 – Future Works

After obtaining the d-patterns(deployed patterns) we need to prune noisy patterns using closed sequential pattern and inner pattern evolution by calculating the term weights based on their occurrence in the discovered patterns.

Also develop another model for preprocessing of text file that is removing stop words, punctuation and tokenizing the terms.

12.1 INNER PATTERN EVOLUTION

We discuss how to reshuffle supports of terms within normal forms of d-patterns based on negative documents in the training set. The technique will be useful to reduce the side effects of noisy patterns because of the low- frequency problem. This technique is called inner pattern evolution here, because it only changes a pattern's term supports within the pattern. A threshold is usually used to classify documents into relevant or irrelevant categories. The proposed model includes two phases: the training phase and the testing phase. In the training phase, the proposed model first calls Algorithm PTM (Dþ, min sup) to find d-patterns in positive documents (Dþ) based on a min sup, and evaluates term supports by deploying d-patterns to terms. It also calls Algorithm IPEvolving to revise term supports using noise negative documents in D_ based on an experimental coefficient. The incoming documents then can be sorted based on these weights.

12.1 Text Preprocessing

Frequently the texts we have are not those we want to analyze. We may have an single file containing the collected works of an author although we are only interested in a single work. Or we may be given a large work broken up into volumes (this is the case for *Les Misèrables*, as we will see later) where the division into volumes is not important to us.

If we are interested in an author's style, we likely want to break up a long text (such as a book-length work) into smaller chunks so we can get a sense of the variability in an author's writing. If we are comparing one group of writers to a second group, we may wish to aggregate information about writers belonging to the same group. This will require merging documents or other information that were initially separate. This section illustrates these two common preprocessing step: splitting long texts into smaller "chunks" and aggregating texts together.

Another important preprocessing step is tokenization. This is the process of splitting a text into individual words or sequences of words (*n-grams*). Decisions regarding tokenization will depend on the language(s) being studied and the research question. For example, should the phrase "**her father's arm-chair**" be tokenized as as **["her", "father", "s", "arm", "chair"]** or ["**her", "father's", "arm-chair"]**.**

Tokenization patterns that work for one language may not be appropriate for another (What is the appropriate tokenization of "Qu'est-ce que c'est?"?). This section begins with a brief discussion of tokenization before covering splitting and merging texts.

Chapter 13 - References, IEEE Format

Research Paper

[1] N. Zhong, Y. Li, and S.T. Wu. Effective pattern discovery for text mining. IEEE Transactions on Knowledge and Data Engineering, DOI: http://doi.ieeecomputersociety.org/10.1109/TKDE.2010.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In Proceedings of the 20th International Conference on Very Large Data Bases, pages 478–499, 1994.

[3] Rohini Y.Thombare, Shirish. S. Sane, Effective Pattern Deploying Approach in Pattern Taxonomy Model for Text Mining, *International Journal of Computer Application*,6,2013

Web References

[4](n.d.). Retrieved from http://en.wikipedia.org/wiki/Data_mining

[5](n.d.). Retrieved from http://www.researchgate.net/publication/220072060_Effective_Pattern_Discovery_for_Text_Mining

[6] (n.d.). Retrieved from http://stackoverflow.com/questions/16145448/diffference-between-closed-and-open-sequential-pattern-mining-algorithms

[7](n.d.). Retrieved from http://en.wikipedia.org/wiki/Text_mining