Preprocessing Techniques for Text Mining - An Overview

Dr. S. Vijayarani[1], Ms. J. Ilamathi[2], Ms. Nithya[3]

Assistant Professor[1], M. Phil Research Scholar[2, 3]

Department of Computer Science, School of Computer Science and Engineering,

Bharathiar University, Coimbatore, Tamilnadu, India[1, 2, 3]

Abstract

Data mining is used for finding the useful information from the large amount of data. Data mining techniques are used to implement and solve different types of research problems. The research related areas in data mining are text mining, web mining, image mining, sequential pattern mining, spatial mining, medical mining, multimedia mining, structure mining and graph mining. This paper discussed about the text mining and its preprocessing techniques. Text mining is the process of mining the useful information from the text documents. It is also called knowledge discovery in text (KDT) or knowledge of intelligent text analysis. Text mining is a technique which extracts information from both structured and unstructured data and also finding patterns. Text mining techniques are used in various types of research domains like natural language processing, information retrieval, text classification and text clustering.

Keywords: Text mining, Stemming, Stop words elimination, TF/IDF algorithms, Word Net, Word Disambiguation.

1.Introduction

Text mining is the process of seeking or extracting the useful information from the textual data. It is an exciting research area as it tries to discover knowledge from unstructured texts. It is also known as Text Data Mining (TDM) and knowledge Discovery in Textual Databases (KDT). KDT plays an increasingly significant role in emerging applications, such as Text Understanding. Text mining process is same as data mining, except, the data mining tools are designed to handle structured data whereas text mining can able to handle unstructured or semi-structured data sets such as emails HTML files and full text documents etc. [1]. Text Mining is used for finding the new, previously unidentified information from different written resources.

Structured data is data that resides in a fixed field within a record or file. This data is contained in relational database and spreadsheets. The unstructured data usually refers to information that does not reside in a traditional row-column database and it is the opposite of structured data. Semi-Structured data is the data that is neither raw data, nor typed data in a conventional database system. Text mining is a new area of computer science research that tries to solve the issues that occur in the area of data mining, machine learning, information extraction, natural language processing, information retrieval, knowledge management and classification. Figure 1 gives the overview of text mining process.

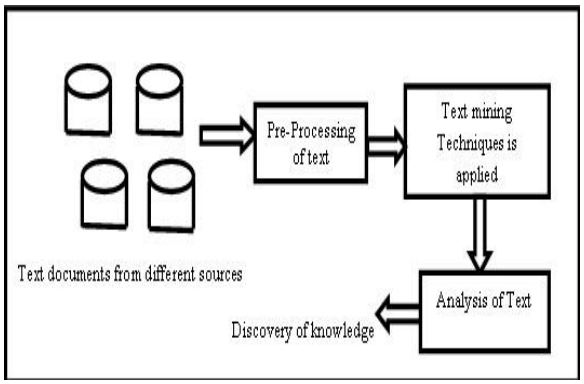

Figure 1. Text Mining Process

The remaining portion of the paper is organized as follows. Section 2 gives the literature review. Section 3 describes the text mining preprocessing methods. Stemming algorithms for classification are discussed in Section 4. Conclusion is given in Section 5.

1.1 Applications of Text Mining

Information Retrieval

Information retrieval (IR) concept has been developed in relation with database systems for many years. Information retrieval is the association and retrieval of information from a large number of text-based documents. The information retrieval and database systems, each handle various kinds of data; some database system problems are usually not present in information retrieval systems, such as concurrency control, recovery, transaction management, and update. Also, some common information retrieval problems are usually not encountered in conventional database systems, such as unstructured documents, estimated search based on keywords, and the concept of relevance. Due to the huge quantity of text information, information retrieval has found many applications. There exist many information retrieval systems, such as on-line library catalog systems, on-line document management systems, and the more recently developed Web search engines [1].

Information Extraction

The information extraction method identifies key words and relationships within the text. It does this by looking for predefined sequences in the text, a process called pattern matching. The software infers the relationships between all the identified places, people, and time to give the user with meaningful information. This technology is very useful when dealing with large volumes of text. Traditional data mining assumes that the information being "mined" is already in the form of a relational database. Unfortunately, for many applications, electronic information is only available in the form of free natural language documents rather than structured databases [1]. This process is depicted in Figure 2.

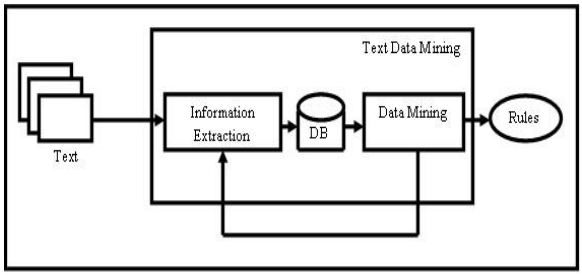

Figure 2. Process of Text Extraction

Categorization

Categorization involves identifying the main themes of a document by inserting the document into a pre-defined set of topics. When categorizing a document, a computer program will often treat the document as a "bag of words." It does not try to process the actual information as information extraction does. Rather, the categorization only counts words that appear and, from the counts, identifies the main topics that the document covers. Categorization often relies on a glossary for which topics are predefined, and relationships are identified by looking for large terms, narrower terms, synonyms, and related terms [4].

Natural Language Processing

Natural Language Processing (NLP) is an area of research and application that explores how computers can be used to understand and manipulate natural language text. NLP researchers aim to collect knowledge on how human beings understand and use language so that fitting tools and techniques can be developed to make computer systems understand and manipulate natural languages to perform the preferred tasks [3].

The basics of NLP lie in a number of disciplines, viz. computer and information sciences, linguistics, mathematics, electrical and electronic engineering, artificial intelligence and robotics, psychology, etc. Applications of NLP include a number of fields of studies, such as machine translation, natural language text processing and summarization, user interfaces, multilingual and cross language information retrieval (CLIR), speech recognition, artificial intelligence and expert systems and so on[3].

2. Literature Review

Anjali Ganesh Jivani [22] discussed that the purpose of stemming is to reduce different grammatical forms or word forms of a word like its noun, adjective, verb, adverb etc. The goal of stemming is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. This paper discusses different methods of stemming and their comparisons in terms of usage, advantages as well as limitations. The basic difference between stemming and lemmatization is also discussed.

Vishal Gupta et.al [23] has analyzed the stemmer's performance and effectiveness in applications such as spelling checker varies across languages. A typical simple stemmer algorithm involves removing suffixes using a list of frequent suffixes, while a more complex one would use morphological knowledge to derive a stem from the words. The paper gives a detailed outline of common stemming techniques and existing stemmers for Indian languages.

K.K. Agbele [24] discussed the technique for developing pervasive computing applications that are flexible and adaptable for users. In this context, however, information retrieval (IR) is often defined in terms of location and delivery of documents to a user to satisfy their information need. In most cases, morphological variants of words have similar semantic interpretations and can be considered as equivalent for the purpose of IR applications. The algorithm Context-Aware Stemming (CAS) is proposed, which is a modified version of the extensively used Porter's stemmer. Considering only generated meaningful stemming words as the stemmer output, the results show that the modified algorithm significantly reduces the error rate of Porter's algorithm from 76.7% to 6.7% without compromising the efficacy of Porter's algorithm.

Hassan Saif [25] has investigated whether removing stop words helps or hampers the effectiveness of Twitter sentiment classification methods. For this investigation he has applied, six different stop word identification methods to Twitter data from six different datasets and observe how removing stop words affects two well-known supervised sentiment classification methods. The result shows that using pre-compiled lists of stop words negatively impacts the performance of Twitter sentiment classification approaches. On the other hand, the dynamic generation of stopword lists, by removing those infrequent terms appearing only once in the corpus appears to be the optimal method for maintaining a high classification performance while reducing the data sparsity and substantially shrinking the feature space.

3. Preprocessing methods

Preprocessing method plays a very important role in text mining techniques and applications. It is the first step in the text mining process. In this paper, we discuss the three key steps of preprocessing namely, stop words removal, stemming and TF/IDF algorithms (Figure 3).

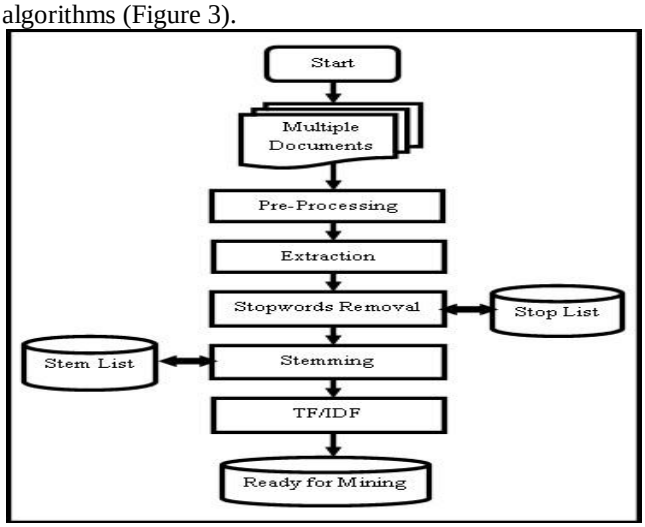

Figure 3. Text Mining Pre-Processing Techniques

A. Extraction

This method is used to tokenize the file content into individual word.

B. Stop Words Elimination

Stop words are a division of natural language. The motive that stop-words should be removed from a text is that they make the text look heavier and less important for analysts. Removing stop words reduces the dimensionality of term space. The most common words in text documents are articles, prepositions, and pro-nouns, etc. that does not give the meaning of the documents. These words are treated as stop words. Example for stop words: the, in, a, an, with, etc. Stop words are removed from documents because those words are not measured as keywords in text mining applications [5].

C. Stop word removal methods

Four types of stop word removal methods are followed, the methods are used to remove stop words from the files [5].

i. The Classic Method: The classic method is based on removing stop words obtained from pre-compiled lists [7].

ii. Methods based on Zipf's Law (Z-Methods): In addition to the classic stop list, we use three stop word creation methods moved by Zipf's law, including: removing most frequent words (TF-High) and removing words that occur once, i.e. singleton words (TF1). We also consider removing words with low inverse document frequency (IDF) [7, 8].

iii. The Mutual Information Method (MI)

The mutual information method (MI) is a supervised method that works by computing the mutual information between a given term and a document class (e.g., positive, negative), providing a suggestion of how much information the term can tell about a given class. Low mutual information suggests that the term has a low discrimination power and consequently it should be removed [7, 8].

iv. Term Based Random Sampling (TBRS)

This method was first proposed by Lo et al. (2005) to manually detect the stop words from web documents. This method works by iterating over separate chunks of data which are randomly selected. It then ranks terms in each chunk based on their in format values using the Kullback-Leibler divergence measure as shown in Equation 1.

$$d_x(t) = P_x(t).\log_2 \frac{Px(t)}{p(t)} \quad (1)$$

Where Px (t) is the normalized term frequency of a term t within a mass x, and P(t) is the normalized term frequency of t in the entire collection. The final stop list is then constructed by taking the least informative terms in all chunks, removing all possible duplications [7].

D. Stemming

This method is used to identify the root/stem of a word. For example, the words connect, connected, connecting, connections all can be stemmed to the word "**connect**" [6]. The purpose of this method is to remove various suffixes, to reduce the number of words, to have accurately matching stems, to save time and memory space. This is illustrated in Figure 4.

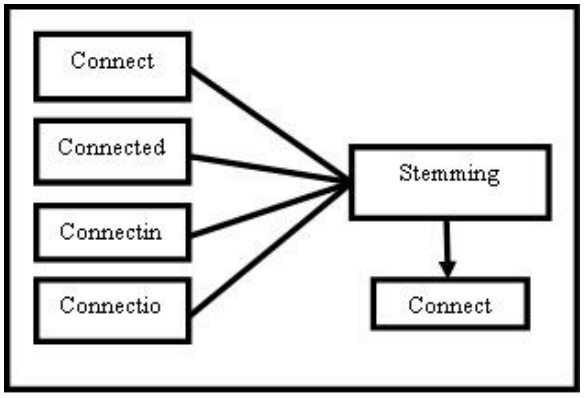

Figure 4. Stemming Process

In stemming, translation of morphological forms of a word to its stem is done assuming each one is semantically related. There are two points are considered while using a stemmer:

- Words that do not have the same meaning should be kept separate
- Morphological forms of a word are assumed to have the same base meaning and hence it should be mapped to the same stem

These two rules are good and sufficient in text mining or language processing applications. Stemming is usually considered as a recall-enhancing device. For languages with relatively simple morphology, the power of stemming is less than for those with a more complex morphology. Most of the stemming experiments done so far are in english and other west European languages.

4. Stemming Algorithms for Classification Process

Usually, stemming algorithms can be classified into three groups: truncating methods, statistical methods, and mixed methods [8]. Each of these groups has a typical way of finding the stems of the word variants. These methods and the algorithms discussed in this paper are shown in the Figure 5.

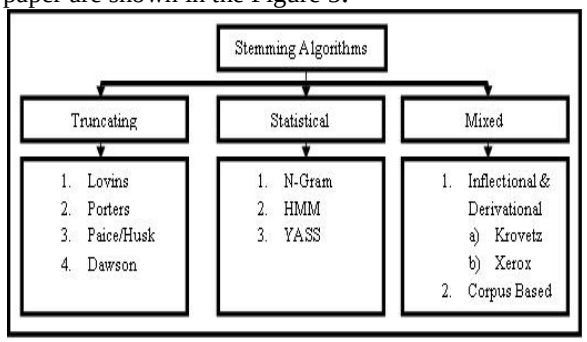

Figure 5. Stemming Algorithms

A. Truncating Methods (Affix Removal)

As the name obviously suggests these methods are related to removing the suffixes or prefixes (commonly known as affixes) of a word [8]. The most basic stemmer is the Truncate (n) stemmer which truncated a word at the nth symbol i.e. keep n letters and remove the rest. In this method words shorter than n are kept as it is. The probability of over stemming increases when the word length is small. Another simple approach was the S-stemmer – an algorithm conflating singular and plural forms of English nouns. This algorithm was proposed by Donna Harman. The algorithm has rules to remove suffixes in plurals so as to convert them to the singular forms [9].

1. Lovins Stemmer

This was the first trendy and effective stemmer proposed by Lovins in 1968. The Lovins stemmer removes the longest suffix from a word. Once the ending is removed, the word is recoded using a different table that makes various adjustments to convert these stems into valid words. It always removes a maximum of one suffix from a word, due to its nature as a single pass algorithm. The advantages of this algorithm are it is very fast and can handle the removal of double letters in words like 'getting' being transformed to 'get' and also handles many irregular plurals like – mouse and mice, index and indices etc. A drawback of the Lovins approach is it is time consuming one. Furthermore, many suffixes are not available in the table of endings. It is sometimes highly unreliable and frequently fails to form words from the stems or to match the stems of like-meaning words.

2. Porters Stemmer

Porters stemming algorithm [11, 12] is one of the most popular stemming algorithm proposed in 1980. Many modifications and enhancements have been made and suggested on the basic algorithm. It is based on the idea that the suffixes in the English language (approximately 1200) are mostly made up of grouping of smaller and simpler suffixes. It has five steps, and within each step, rules are applied until one of them passes the conditions. If a rule is accepted, the suffix is removed consequently, and the next step is performed. The resultant stem at the end of the fifth step is returned.

The rule looks like the following:
<condition> <suffix> → <new suffix>

For example, a rule (m>0) EED → EE means "if the word has at least one vowel and consonant plus EED ending, change the ending to EE". So "agreed" becomes "agree" while "feed" remains unchanged. This algorithm has about 60 rules and very easy to understand. Porter designed a detailed framework of stemming which is known as 'Snowball'. The main purpose of the framework is to allow programmers to develop their own stemmers for other character sets or languages.

However it was noted that Lovins stemmer is a heavier stemmer that produces a better data reduction [13]. The Lovins algorithm is obviously larger than the Porter algorithm, because of its very extensive endings list. But in one way that is used to advantage: it is faster. It has effectively traded space for time, and with its large suffix set it needs just two major steps to remove a suffix, compared with the five of the Porter algorithm.

3. Paice/Husk Stemmer

The Paice/Husk stemmer is an iterative algorithm with one table containing about 120 rules indexed by the last letter of a suffix [14]. It tries to find the applicable rule by the last character of the word. Each rule specifies either a deletion or replacement of an ending. If there is no such rule, it terminates. It also terminates if a word starts with a vowel and there are only two letters left or only three characters left. Otherwise, the rule is applied and the process repeats. The advantage is simple and every iteration taking care of both deletion and replacement as per the rule applied. The disadvantage is it is very heavy algorithm and over stemming may occur.

4. Dawson Stemmer

This stemmer is an extension of the Lovins approach except that it covers much more complete list of about 1200 suffixes. Like Lovins, it is also a single pass stemmer and hence it is pretty fast. The suffixes are stored in the reversed order indexed by their length and last letter. In fact they are organized as a set of divided character trees for rapid access. The advantage is that it covers more suffixes than Lovins and is fast in execution. The disadvantage is it is very complex, and lacks a standard reusable implementation [8].

B. Statistical Methods

These are the stemmers who are based on statistical analysis and techniques. Most of the methods remove the affixes, but after implementing some statistical procedure [8].

1. N-Gram Stemmer

It is language independent stemmer. The string-similarity approach is used to convert word inflation to its stem. N-gram is a string of n, usually adjacent, characters extracted from a section of continuous text. N-gram is a set of n following characters extracted from a word. The main idea behind this approach is, similar words will have a high quantity of n-grams in common. For n equals to 2 or 3, the words extracted are called diagrams or trigrams, respectively [7, 8].

For example, the word 'INTRODUCTIONS' results in the generation of the diagrams:

I, IN, NT, TR, RO, OD, DU, UC, CT, TI, IO, ON, NS, S and the trigrams:
I, *IN, INT, NTR, TRO, ROD, ODU, DUC, UCT, CTI, TIO, ION, ONS, NS*, S

Where '*' denotes a padding space. There are n+1 such diagram and n+2 such trigrams in a word containing n characters. Most stemmers are language-specific. Usually a value of 4 or 5 is selected for n. After that a textual data or document is analyzed for all the n-grams. It is clear that a word root generally occurs less frequently than its morphological form. This means a word generally has an affix associated with it.

This stemmer has an advantage that it is language independent and hence very useful in many applications. The disadvantage is it requires huge memory and storage for creating and storing the n grams and indexes and hence it is not a practical approach.

2. HMM Stemmer

This stemmer is based on the concept of the Hidden Markov Model (HMMs) which are finite-state automata where transitions between states are ruled by probability functions. At each transition, the new state emits a symbol with a given probability. This model was proposed by Melucci and Orio [15]. This method is based on unsupervised learning and does not need a prior linguistic knowledge of the dataset. In this method the probability of each path can be computed and the most probable path is found in the automata graph. In order to apply HMMs to stemming, a sequence of letters that forms a word can be considered the result of a concatenation of two subsequences: a prefix and a suffix. A way to model this process is through an HMM where the states are divided into two disjoint sets: initial can be the stems only and the latter can be the stems or suffixes. Transitions between states define word structure process. There are some assumptions that can be made with this method:

1. Initial states belong only to the stem-set – a word always starts with a stem
2. Transitions from states of the suffix-set of states of the stem-set always have a null probability - a word can be only a concatenation of a stem and a suffix.
3. Final states belong to both sets - a stem can have a number of different derivations, but it may also have no suffix.

The advantage of this method is it is unsupervised and hence knowledge of the language is not required. The disadvantage is it is a little complex and may over stem the words sometimes [15].

3. YASS Stemmer

The name is an acronym for Yet another Suffix Striper. This stemmer was proposed by Prasenjit Majumder [16]. According to the authors the performance of a stemmer generated by clustering a lexicon without any linguistic input is equivalent to that obtained using standard, rule-based stemmers such as Porter's. This stemmer comes under the class of statistical as well as corpus based. It does not rely on linguistic expertise. Retrieval experiments by the authors in French, English, and Bengali datasets which shows that the proposed approach is effective for languages that are primarily suffixed in nature.

C. Mixed Methods

1. Inflectional and Derivational Methods

This is another approach in stemming and it involves both the inflectional as well as the derivational morphology analysis. The corpus should be very large to develop these types of stemmers and hence they are part of corpus base stemmers too. In case of inflectional the word variants are related to the language specific syntactic variations like a plural, gender, case, etc., whereas in derivational the word variants are related to the part-of-speech (POS) of a sentence where the word occurs [7].

a. Krovetz Stemmer (KSTEM)

The Krovetz stemmer was presented in 1993 by Robert Krovetz [17] and is a linguistic lexical validation stemmer. Since it is based on the inflectional property of words and the language syntax, it is very complicated in nature. It effectively and accurately removes inflectional suffixes in three steps:

1. Transforming the plurals of a word to its singular form

2. Converting the past tense of a word to its present tense

3. Removing the suffix 'ing'

The conversion process first removes the suffix and then through the process of checking in a dictionary for any recoding, returns the stem to a word. The dictionary lookup also performs any transformations that are required due to spelling exception and also converts any stem produced into a real word. Since this stemmer does not find the stems for all word variants, it can be used as a pre stemmer before actually applying a stemming algorithm. This would increase the speed and effectiveness of the main stemmer. Compared to Porter and Paice / Husk, this is a very light stemmer.

The Krovetz stemmer attempts to increase accuracy and robustness by treating spelling errors and meaningless stems. If the input document size is large this stemmer becomes weak and does not perform very effectively. The major and clear flaw in dictionary-based algorithms is their incapability to manage with words, which are not in the lexicon. This stemmer does not consistently produce a good recall and precision performance [17].

b. Xerox Inflectional and Derivational Analyzer

The linguistics groups at Xerox have developed a number of linguistic tools for English which can be used in information retrieval. In particular, they have produced an English lexical database which provides a morphological analysis of any word in the lexicon and identifies the base form. Xerox linguists have developed a lexical database for English and some other languages also which can analyze and generate inflectional and derivational morphology. The inflectional database reduces each surface word to the form which can be found in the dictionary, as follows [12]:

- nouns singular (e.g. children child)
- verbs infinitive (e.g. understood understand)
- adjectives positive form (e.g. best good)
- pronoun nominative (e.g. whom who)

The derivational database reduces surface forms to stems which are related to the original in both form and semantics.

Advantages

This stemmer works well with a large document also and removes the prefixes also whereever applicable. All stems are valid words since a lexical database which provides a morphological analysis of any word in the lexicon is available for stemming. It has proved to work better than the Krovetz stemmer for a large corpus.

Disadvantage

The output depends on the lexical database which may not be exhaustive. Since this method is based on a lexicon, it cannot correctly stem the words which are not part of the lexicon. This stemmer has not been implemented successfully in many other languages. Dependence on the lexicon makes it a language dependent stemmer.

2. Corpus Based Stemmer

This method of stemming was proposed by Xu and Croft in their paper "Corpus-based stemming using co-occurrence of word variants" [8]. They have optional an approach which tries to overcome some of the drawbacks of Porter stemmer.

For example, the words 'policy' and 'police' are conflated though they have a different meaning, but the word 'index' and 'indices' are not conflated though they have the same root. Porter stemmer also generates stems which are not real words like 'iteration' becomes 'iter' and 'general' becomes 'gener'.

Corpus based stemming refers to automatic modification of conflation classes – words that have resulted in a common stem, to suit the characteristics of a given text corpus using statistical methods. The advantage of this method is it can potentially avoid making conflations that are not appropriate for a given corpus and the result is an actual word and not an incomplete stem. The disadvantage is that we need to develop the statistical measure for every corpus separately and the processing time increases as in the first step two stemming algorithms are first used before using this method.

3. Context Sensitive Stemmer

This is a very interesting method of stemming unlike the usual method where stemming is done before indexing a document, over here for a Web Search, context sensitive analysis is done using statistical modeling on the query side. This method was proposed by Funchun Peng et. al.[19].

Basically for the words of the input query, the morphological variants which would be useful for the search are predicted before the query is submitted to the search engine. This severely reduces the number of bad expansions, which in turn reduces the cost of additional computation and improves the precision at the same time. After the predicted word variants of the query have been derived, a context sensitive document matching is done for these variants. This conservative strategy serves as a safeguard against spurious stemming, and it turns out to be very important for improving precision. This stemming

process is divided into four steps [19] after the query is fired:

a. Candidate generation:

Over here the Porter stemmer is used to generate the stems from the query words. This has completely no relation to the semantics of the words. For a better output the corpus-based analysis based on distributional similarity is used. The foundation of using distributional word similarity is that true variants tend to be used in similar contexts. In the distributional word similarity calculation, each word is represented by a vector of features derived from the context of the word. We use the bigrams to the left and right of the word as its context features, by mining a huge Web corpus. The similarity between two words is the cosine similarity between the two corresponding feature vectors [7].

b. Query Segmentation and head word detection:

When the queries are long, it is important to detect the major concept of the query. The query is broken into segments which are normally the noun phrases. For each noun phrase the most important word is detected which is the head word. Sometimes a word is split to know the content. The mutual information of two adjacent words is found and if it passes a threshold value, they are kept in the same segment. Finding the headword is by using a syntactical parser [7, 8].

c. Context sensitive word expansion:

The keywords words are obtained by using probability measures and it decided which word variants would be most useful – generally they are the plural forms of the words. This is done using the simplest and most successful approach to language modeling, which is the one based on the *n*-gram model which uses the chain rule of probability. In this step all the important head word variants are obtained. The traditional way of using stemming for Web search, is referred as the naïve model. This is to treat every word variant equivalent for all possible words in the query. The query "book store" will be transformed into "*(book OR books) (store OR stores)*" when limiting stemming to pluralization handling only, where *OR* is an operator that denotes the equivalence of the left and right arguments [8].

d. Context sensitive document matching:

The context is the left or the right non-stop segments of the original word. Considering the fact that queries and documents may not represent the intent in exactly the same way, this proximity constraint is to allow variant occurrences within a window of some fixed size [7]. The smaller the window size is, the more restrictive the matching. The advantage of this stemmer is it improves selective word expansion on the query side and conservative word occurrence matching on the document side. The disadvantage is the processing time and the complex nature of the stemmer. There can be errors in finding the noun phrases in the query and the nearest words.

Term Frequency-Inverse Document Frequency

Term Frequency–Inverse Document Frequency (tf-idf) is a numerical statistic which reveals that a word is how important to a document in a collection. The Tf - IDF is often used as a weighting factor in information retrieval and text mining. The value of tf-idf increases proportionally to the number of times a word appears in the document, but is counteracting by the frequency of the word in the corpus. This can help to control the fact that some words are generally more common than others. Tf–IDF can be successfully used for stop-words filtering in various subject fields including text summarization and classification. Tf–IDF is the product of two statistics which are termed frequency and inverse document frequency. To further distinguish them, the number of times each term occurs in each document is counted and sums them all together. Term Frequency (TF) is defined as the number of times a term occurs in a document [20, 21].

$$Tf(t,d) = .5 + \frac{0.5 * f(t,d)}{Masimum\ \ occurences\ \ of\ words}$$

Inverse Document Frequency- an Inverse Document Frequency (IDF) is a statistical weight used for measuring the importance of a term in a text document collection. IDF feature is incorporated which reduces the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely.

$$\text{IDF (t, d)} log\ = \frac{\text{lDl}}{(no.of\ doc., term\ t\ appearns)}$$

Then Term Frequency - Inverse document frequency [TF-IDF] is calculated for each word using the formula,

$$\text{Tfidf (t, f, d)} = \text{tf (t, d)} * \text{idf (t, d)}$$

In this equation (1) and (2) ft, d denotes the frequency of the occurrence of term t in document d. In equation (3) TF-IDF is calculated for each term in the document by using Term Frequency (Tft, d) and Inverse Document Frequency (idft, d).

4. Conclusion

Text mining is the process of seeking or extracting the useful information from the textual data. It tries to find interesting patterns from large databases. It uses different pre-processing techniques likes stop words elimination and stemming. This paper has given complete information about the text mining preprocessing techniques, i.e. stop words elimination and stemming algorithms. We hope this paper will help the text mining researcher's community and they get good knowledge about various preprocessing techniques.

Reference

[1] Vishal Gupta and Gurpreet S. Lehal, A Survey of Text Mining Techniques and Applications, JOURNAL OF EMERGING TECHNOLOGIES IN WEB INTELLIGENCE, VOL. 1, NO. 1, AUGUST 2009.

[2] Shaidah Jusoh 1and Hejab M. Alfawareh, Techniques, Applications and Challenging Issue in Text Mining, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 6, No 2, November 2012, ISSN (Online): 1694-0814.

[3] S.Jusoh and H.M. Alfawareh, Natural language interface for online sales, in Proceedings of the International
Conference on Intelligent and Advanced System (ICIAS2007). Malaysia: IEEE, November 2007, pp. 224–228.

[4] Saleh Alsaleem, Automated Arabic Text Categorization Using SVM and NB, International Arab Journal of e-Technology, Vol. 2, No. 2, June 2011.

[5] M.F. Porter, An Algorithm for Suffix Stripping, Program, vol. 14, no. 3, pp. 130-137, 1980.

[6] C.Ramasubramanian and R.Ramya, Effective Pre-Processing Activities in Text Mining using Improved Porter's Stemming Algorithm, International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 12, December 2013, ISSN (Online) : 2278-1021.

[7] Ms. Anjali Ganesh Jivani, A Comparative Study of Stemming Algorithms, Anjali Ganesh Jivani et al, Int. J. Comp. Tech. Appl., Vol 2 (6), 1930-1938, ISSN:2229-6093.

[8] Deepika Sharma, Stemming Algorithms, A Comparative Study and their Analysis, International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868, Foundation of Computer Science FCS, New York, USA, Volume 4– No.3, September 2012 – www.ijais.org.

[9] Harman Donna, How effective is suffixing? Journal of the American Society for Information Science, 1991; 42, 7-15 7.

[10] J. B. Lovins, Development of a stemming algorithm, Mechanical Translation and Computer Linguistic., vol.11, no.1/2, pp. 22-31, 1968.

[11] Porter M.F, An algorithm for suffix stripping, Program. 1980; 14, 130-137.

[12] Porter M.F, Snowball: A language for stemming algorithms. 2001.

[13] Mladenic Dunja, Automatic word lemmatization. Proceedings B of the 5th International Multi- Conference Information Society IS. 2002, 153-159.

[14] Paice Chris D, Another stemmer. ACM SIGIR Forum, Volume 24, No. 3. 1990, 56-61.

[15] Melucci Massimo and Orio Nicola, A novel method for stemmer generation based on hidden Markov models. Proceedings of the twelfth international conference on Information and knowledge management. 2003, 131-138.

[16] Plisson Joel, Lavrac Nada and Mladenic Dunja, A rule based approach to word lemmatization. Proceedings C of the 7th International Multi-Conference Information Society IS. 2004.

[17] Krovetz Robert, Viewing morphology as an inference process. Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval. 1993,191-202.

[18] Xu Jinxi and Croft Bruce W, Corpus-based stemming using co-occurrence of word variants.
ACM Transactions on Information Systems. Volume 16, Issue 1. 1998, 61-81.

[19] Funchun Peng, Nawaaz Ahmed, Xin Li and Yumao Lu, Context sensitive stemming for web search.
Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. 2007, 639-646.

[20] Menaka S and Radha N, Text Classification using Keyword Extraction Technique, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 12, December 2013, ISSN: 2277 128X.

[21] S.Charanyaa and K.Sangeetha, Term Frequency Based Sequence Generation Algorithm for Graph Based Data Anonymization, International Journal of Innovative Research in Computer and Communication Engineering, (An ISO 3297: 2007 Certified Organization), Vol. 2, Issue 2, February 2014, ISSN(Online): 2320-9801.

[22] Anjali Ganesh Jivani , A Comparative Study of Stemming Algorithms, International Journal of Computer, Technology and Application, Volume 2, ISSN:2229-6093.

[23] Vishal Gupta, Gurpreet Singh Lehal, A Survey of Common Stemming Techniques and Existing Stemmers for Indian Languages, Journal of Emerging Techniloigies in Web Intelligence, VOL. 5, NO. 2, MAY 2013.

[24] Agbele, A.O. Adesina, N.A. Azeez, & A.P. Abidoye , Context-Aware Stemming Algorithm for Semantically Related Root Words, African Journal of Computing & ICT.

[25] Hassan Saif, Miriam Fernandez,Yulan He, Harith Alani , On Stopwords, Filtering and Data Sparsity for Sentiment Analysis of Twitter.