Chapter 3: Text Pre-Processing

This chapter starts with introduction about pre-processing followed by steps in preprocessing. Each pre-processing step is discussed in detail. The pre-processing method and technique implemented in the proposed work is also deliberated upon. Finally, the chapter ends with evaluation of the summary with several metrics.

3.1 Introduction

Though this is considered the preliminary step to be conducted before actually applying Text Mining algorithms/methods, it is a very important process and this routine itself is divided into a number of sub-methods, which again have optional algorithms with their own set of advantages and disadvantages. The text data on which I have executed the algorithm have been first converted to text format if it was not so. In fact, the majority of the datasets were already in text format.

Most of the Text Mining approaches are based on the idea that a text document can be described on the set of words contained in it i.e. bag-of-words representation. The pre-processing itself is made up of a sequence of steps. The steps are explained in detail.

3.2 Morphological Analysis

The first step in text-preprocessing is the morphological analyses. It is divided into three subcategories: tokenization, filtering and stemming. Morphology is a part of linguistics which is dealing with words. Therefore, it deals with the smallest, useful unit of a document. One could say that characters are the smallest unit. Nonetheless, characters do not carry any valuable information for information retrieval. Firstly, Text Mining requires the words and the endings of a document. Finding words and separating them is known as tokenization.
The next step is filtering of important and relevant words from our list of words which were the output of tokenization. This is also called stop words removal.

The third step is stemming. Stemming is very important and a lot of research work has already been done on it. Stemming reduces words variants to its root form. Stemming of words increases the recall and precision of the information retrieval in Text Mining. The term recall describes the proportion of all relevant documents in a data set that are retrieved by the information retrieval system. The term precision describes the proportion of relevant documents in the data set returned to the user. Precision and recall are two very important measures for text categorization, clustering as well as summarization. The details are discussed further as and when they are applied.

3.3 Tokenization

Over here, the input document is split into a set of words by removing all punctuation marks, tabs and other non-text characters and replacing them with white spaces. The part-of-speech (POS) tagging is also applied in some cases where words are tagged according to the grammatical context of the word in the sentence, hence dividing the words into nouns, verbs, etc. This is important for the exact analysis of relations between words.

Another approach was to ignore the order in which the words occurred and instead focus on their statistical distributions (the bag-of-words approach). In this case, it is necessary to index the text into data vectors. I have used the bag-of-words approach in implementing the algorithms. The POS becomes important if the research is related to NLP. In one algorithm as part of extension work POS has been implemented.

Tokenization has been done using Visual Basic (using strip () function) as well as Matlab (using strtok () function). The Matlab function was found to be much more efficient and fast.

3.4 Filtering

This step is related to removing words, which are of no importance for our Text Mining process like articles, prepositions, conjunctions, etc. This is known as ‘Stop Words Filtering’. It is controlled by human input and not automated. There is not one definite list of stop words, which all tools use, if even used. The stop words list is available on the site of the Onix Text Retrievel Tookit and the site is:

http://www.lextek.com/manuals/onix/stopwords1.html.
This is a very popular list and as per the requirement, the list can be modified. I have used this list to remove the stop words. Another popular list is available on the MIT site and can be downloaded from:
http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop.

3.5 Stemming

3.5.1 Introduction to Stemming

Word stemming is an important feature supported by present day indexing and search systems. Indexing and searching are in turn part of Text Mining applications, Natural Language Processing (NLP) systems and Information Retrieval (IR) systems. The main idea is to improve recall by automatic handling of word endings by reducing the words to their word roots, at the time of indexing and searching. Recall in increased without compromising on the precision of the documents fetched. Stemming is usually done by removing any attached suffixes and prefixes (affixes) from index terms before the actual assignment of the term to the index. Since the stem of a term represents a broader concept than the original term, the stemming process eventually increases the number of retrieved documents in an IR system. Text clustering, categorization and summarization also require this conversion as part of the pre-processing before actually applying any related algorithm.

Errors in Stemming

There are mainly two errors in stemming – over stemming and under stemming. Over-stemming is when two words with different stems are stemmed to the same root. This is also known as a false positive. Under-stemming is when two words that should be stemmed to the same root are not. This is also known as a false negative. Paice has proved that light-stemming reduces the over-stemming errors but increases the under-stemming errors. On the other hand, heavy stemmers reduce the under-stemming errors while increasing the over-stemming errors.

Classification of Stemming Algorithms

Broadly, stemming algorithms can be classified in three groups: truncating methods, statistical methods, and mixed methods. Each of these groups has a typical way of finding the stems of the word variants. These methods are shown in the Figure 3.1.

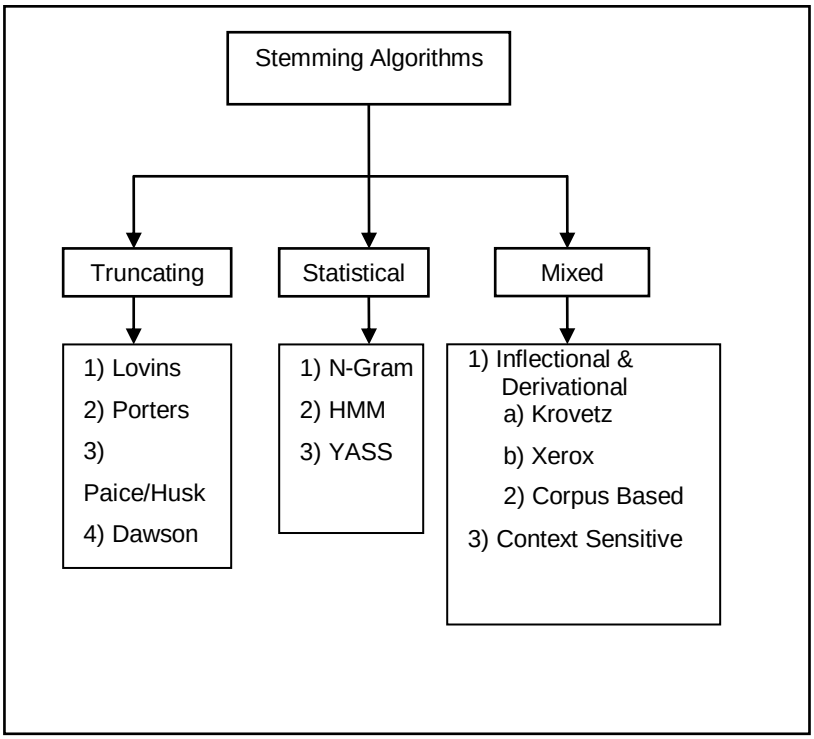

Figure 3-1 Types of Stemming Algorithms

In the work carried out in this research for both the models the Porters Stemmer has been implemented in Python for stemming.

Porters Stemmer

Porters stemming algorithm is as of now one of the most popular stemming methods proposed in 1980. Many modifications and enhancements have been done 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 a combination 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 accordingly, 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 is very easy to comprehend.

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. Currently there are implementations for many Romance, Germanic, Uralic and Scandinavian languages as well as English, Russian and Turkish languages.
Based on the stemming errors, Paice reached to a conclusion that the Porter stemmer produces less error rate than the Lovins stemmer does. However, it was noted that Lovins stemmer is a heavier stemmer that produces a better data reduction. The Lovins algorithm is noticeably bigger than the Porter algorithm, because of its very extensive endings list. Nevertheless, 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.

3.5.2 Stemming and Lemmatizing

The basic function of both the methods – stemming and lemmatizing is similar. Both of them reduce a word variant to its ‘stem’ in stemming and ‘lemma’ in lemmatizing. There is a very subtle difference between both the concepts. In stemming the ‘stem’ is obtaining after applying a set of rules but without bothering about the part of speech (POS) or the context of the word occurrence. In contrast, lemmatizing deals with obtaining the ‘lemma’ of a word, which involves reducing the word forms to its root, form after understanding the POS and the context of the word in the given sentence.

In stemming, conversion of morphological forms of a word to its stem is done assuming each one is semantically related. The stem need not be an existing word in the dictionary but all its variants should map to this form after the stemming has been completed. There are two points to be considered while using a stemmer:

- Morphological forms of a word are assumed to have the same base meaning and hence should be mapped to the same stem

- Words that do not have the same meaning should be kept separate

These two rules are good enough as long as the resultant stems are useful for our Text Mining or language processing applications. Stemming is generally considered as a recall-enhancing device. For languages with relatively simple morphology, the influence of stemming is less than for those with a more complex morphology. Most of the stemming experiments done so far are for English and other west European languages.

Lemmatizing deals with the complex process of first understanding the context, then determining the POS of a word in a sentence and then finally finding the ‘lemma’. In fact, an algorithm that converts a word to its linguistically correct root is called a lemmatizer. A lemma in morphology is the canonical form of a lexeme. Lexeme, in this context, refers to the set of all the forms that have the same meaning, and lemma refers to the particular form that is chosen by convention to represent the lexeme.

In computational linguistics, a stem is the part of the word that never changes even when morphologically inflected, whilst a lemma is the base form of the verb. Stemmers are typically easier to implement and run faster, and the reduced accuracy may not matter for some applications. Lemmatizers are difficult to implement because they are related to the semantics and the POS of a sentence. Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes. The results are not always morphologically right forms of words. Nevertheless, since document index and queries are stemmed "invisibly" for a user, this peculiarity should not be considered as a flaw, but rather as a feature distinguishing stemming from lemmatization. Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the lemma.

For example, the word inflations like gone, goes, going will map to the stem ‘go’. The word ‘went’ will not map to the same stem. However a lemmatizer will map even the word ‘went’ to the lemma ‘go’.

Stemming:

introduction, introducing, introduces – introduc

gone, going, goes – go

Lemmatizing:
introduction, introducing, introduces – introduce
gone, going, goes, went – go

3.6 Syntactical and Semantical Analysis

3.6.1 Syntactical Analysis

This analysis deals with the syntax of a sentence in natural language and is useful in Information Retrieval systems. It can be divided in three parts: part-of-speech tagging, phrase recognition and parsing.

1. Part-of-speech tagging: The recognition of the elements of a sentence like nouns, verbs, adjectives, prepositions, etc. is realized through part of speech tagging (POS tagging).

   The part-of-speech (POS) tagging is also applied in some cases where words are tagged according to the grammatical context of the word in the sentence, hence dividing up the words into nouns, verbs, etc. This is important for the exact analysis of relations between words.
2. Phrase Recognition (PR): This is also very similar to POS. It is required to locate group of words or phrases. PR finds phrases like those given below:
   - Preposition phrase (e.g. in love)
   - Noun Phrase(e.g. the magician of Mecca)
   - Verb Phrase (e.g. do business)
   - Adjectival Phrase (e.g. small house)
   - Adverbial Phrase (e.g. very quickly)
3. Parsing: This process is also part of POS as well as phrase recognition. The sentences are fractionalized into grammatical units. The Stanford parser is very popular for parsing. It generates a tree which is useful for information extraction.

3.6.2 Semantical Analysis

This part of pre-processing deals with the meaning of the textual data i.e. the semantics. It is more or less related to Natural Language Processing.

3.7 The Vector Space Model (VSM)

3.7.1 Introduction to VSM

This model was proposed by Salton and it incorporates the local as well as global information about terms in a document and corpus.

It is an algebraic model for representing text documents as vectors of identifiers. The vector space model procedure can be divided in to three stages. The first stage is the document indexing where content bearing terms are extracted from the document text. The second stage is the weighting of the indexed terms to enhance retrieval of document relevant to the user. The last stage ranks the document with respect to the query according to a similarity measure. The term ‘query’ is used because this model is used in Information Retrieval also.

The similarity between documents or a query and a document is determined through calculations of the cosine similarity, Dice’s coefficient, the Jaccard’s coefficient and in some cases the Eucliean distance. The vector space model has been shown diagrammatically as in Figure 3.1. In the figure, d1 and d2 are document vectors and q1 is the query vector. We call them vectors because they are made up of different terms.

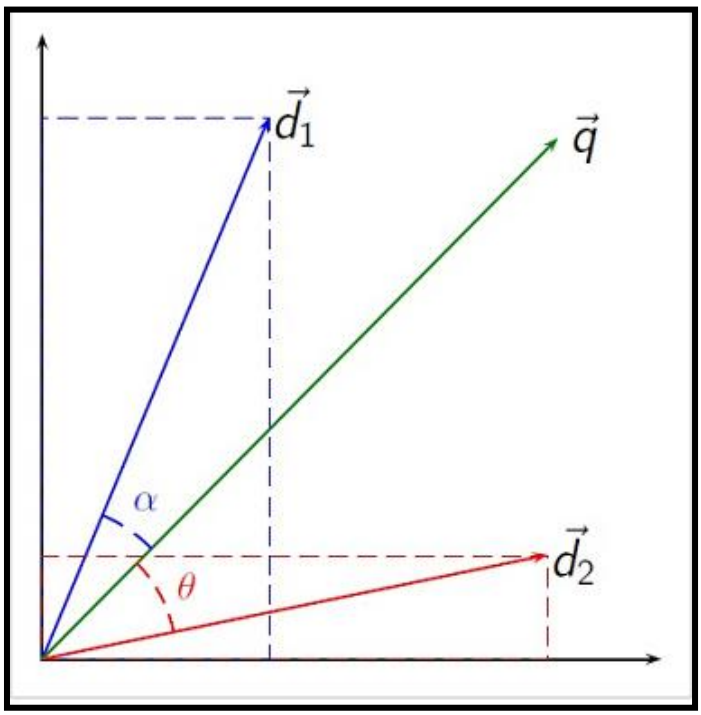

Figure 3-2 The Vector Space Model

The angle between the documents or the query and documents determines the similarity between them.

3.7.2 Term Frequency

The term Frequency method is widely being used for information retrieval systems and text summarization due to its flexibility. The weight is used to measure the importance of word in the document. The measure of importance increases proportionally as the word appears in the document.

The problem with this method is that the terms occurring most frequently might not characterize or contain essentials of the document, as they could be possibly stop words of the document.

TF (Term Frequency) = (No. of times word/terms appear in a document) / (Total no. of words/terms in document)

Luhn (1958) demonstrated the idea of upper and lower cut-off terms for resolving issue regarding the power of significant words.

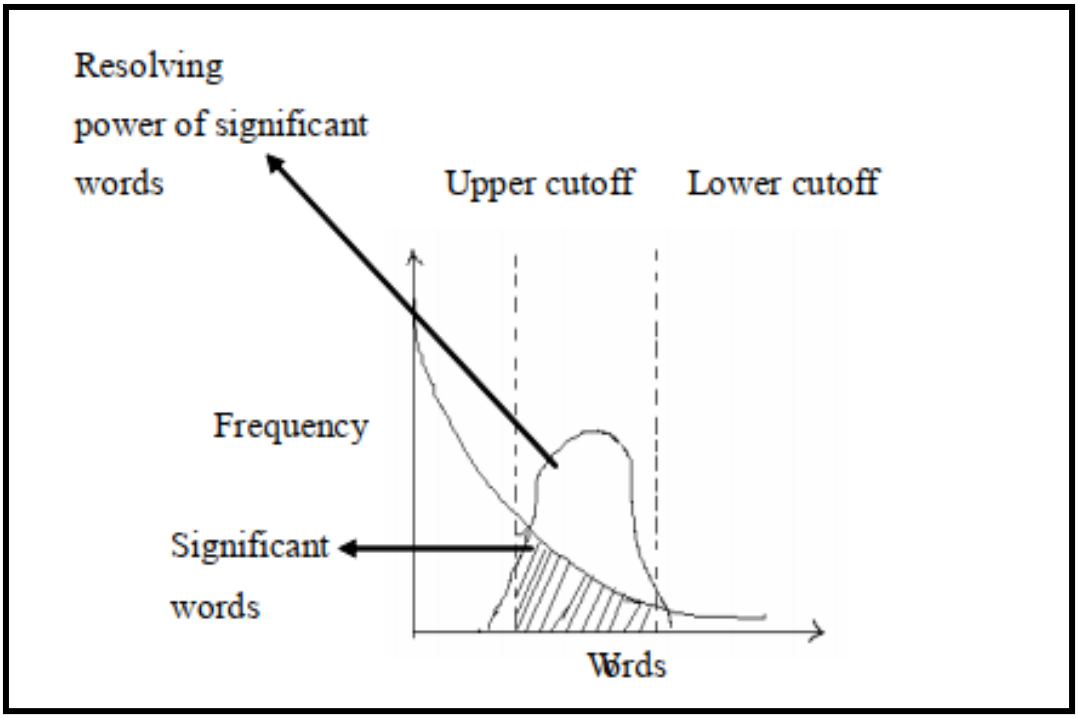

Figure 3-3 Word frequency diagram(S, 2018)

As shown in figure, upper cut-off shows stop words, whereas lower cut-off shows less frequently occurring words in the document. The figure demonstrates the Zipf's law (Zipf 1949) which tells that the frequency of the word is inversely proportional to its rank. Words having score more than the upper cut-off are considered as too common words and the words below the cut-off are considered

as rare terms. These words are not considered as part of summary as they do not exhibit any importance to the content.

3.7.3 The tf-idf score

The document indexing is done using the tf-idf method. It stands for term-frequency (tf) and inverse document frequency (idf). It is weight based on statistics which is assigned to a word to evaluate its importance in a single document or a collection of documents. This weight is also used to generate ranking in documents. It is used in almost all Text Mining algorithms. Over here the assumption is that the first three steps of data pre-processing – tokenization, removing stop words and stemming is already complete.

In the VSM, each document d is considered to be a vector in the term-space i.e. terms that make the document. A document d can be represented as,

$dtf = (tf_1, tf_2, \ldots, tf_n)$,

Where tf_i is the frequency of the i^{th} term in document d. In this way,the *tf* vector can represent each term in a document. Since all documents are not of the same size, we normalize the term frequency by dividing it by the total number of unique terms in the document.

The inverse document frequency (idf) is a measure of the general importance of the term in the corpus. It is obtained by dividing the total number of documents by the number of documents containing the term and taking the logarithm of that quotient.

$$idf(t) = log\ \frac{|\ D\ |}{|\ d{:}\ t \epsilon d\ |} \qquad (3.1)$$

Where,

|D| - total number of documents in the corpus

|d:tϵd| - number of documents where term t appears

If a term is not in the corpus this will lead to division by zero and so we adjust (1) by adding 1 to the denominator. i.e. 1 + |d:tϵd|.

So now the tf-idf score for a term in a document becomes,

$$tf\text{-}idf\ (t, d) = tf \times idf \qquad (3.2)$$

A high weight in tf-idf is reached by a high term frequency in a document and a low document frequency of the term in the whole collection of documents. This will filter out the common terms across the corpus. For terms of more importance in certain algorithms, weights are also assigned i.e. tf score of important terms is multiplied by some integer to increase its weightage.

The tf-idf scoring is very effectively shown in Figure 3.4. For each term of each document in the corpus, in this way the tf-idf score is obtained. A matrix is created to store these scores and then in the Text Mining algorithms these scores are applied. The matrix looks like the example shown in Table 3.1. The actual scores are stored in text files and the Matlab or Visual Basic programs first creates the file and then reads and uses the scores in the programs for the different Text Mining algorithms implementation.

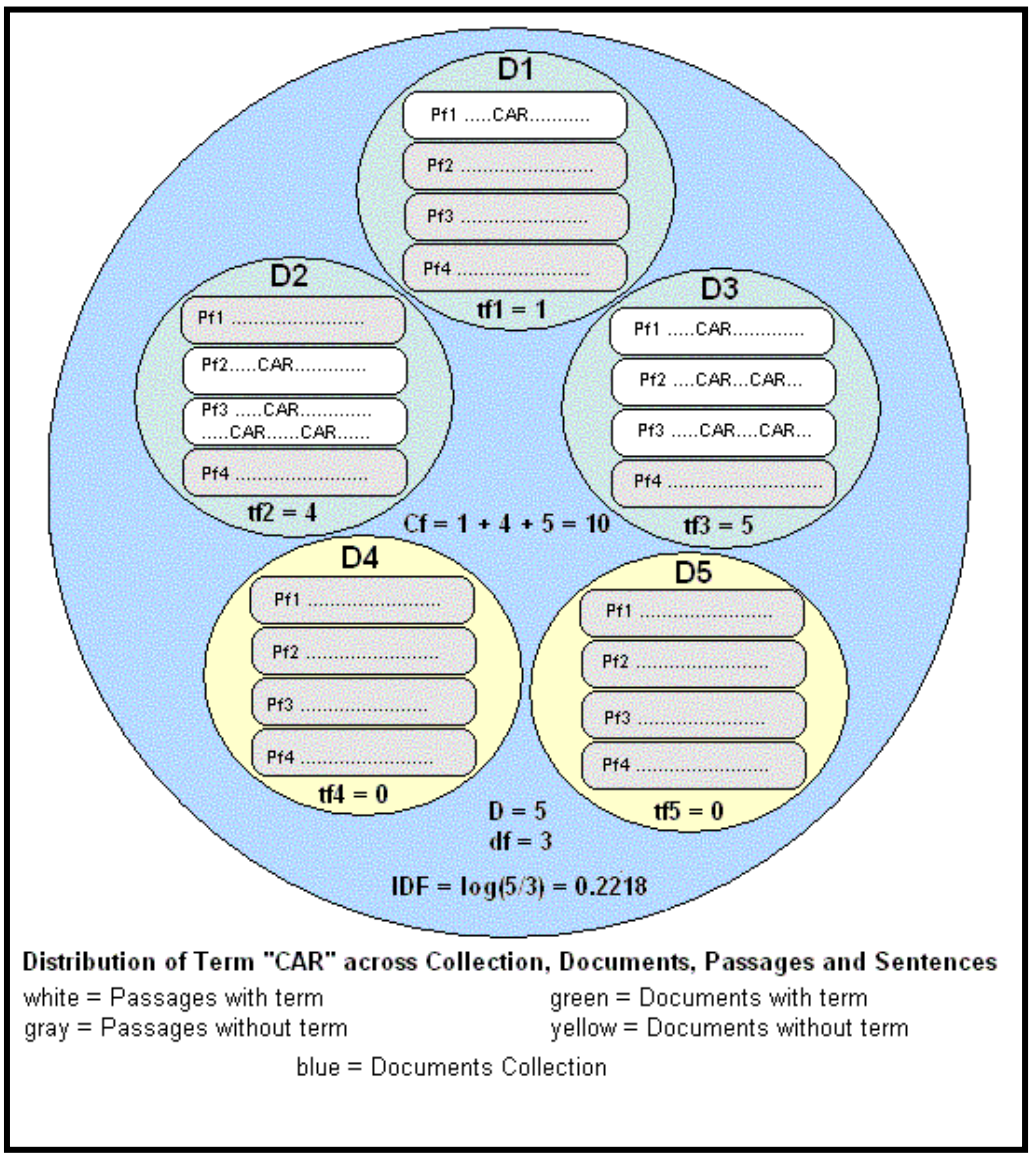

Figure 3-4The term and document frequencies

As shown in the Figure 3-4, the corpus is a collection of documents, documents consist of passages and passages consist of sentences. Thus, for a term *i* in a document *j* we can talk in terms of collection frequencies (Cf), term frequencies (tf), passage frequencies (Pf) and sentence frequencies (Sf).

Table 3-1. The tf-idf matrix example

TERM VECTOR MODEL BASED ON $w_i = tf_i * IDF_i$											
Query, Q: "gold silver truck" D_1: "Shipment of gold damaged in a fire" D_2: "Delivery of silver arrived in a silver truck" D_3: "Shipment of gold arrived in a truck" D = 3; IDF = log(D/df_i)											
		Counts, tf_i						Weights, $w_i = tf_i * IDF_i$			
Terms	Q	D_1	D_2	D_3	df_i	D/df_i	IDF_i	Q	D_1	D_2	D_3
a	0	1	1	1	3	3/3 = 1	0	0	0	0	0
arrived	0	0	1	1	2	3/2 = 1.5	0.1761	0	0	0.1761	0.1761
damaged	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
delivery	0	0	1	0	1	3/1 = 3	0.4771	0	0	0.4771	0
fire	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
gold	1	1	0	1	2	3/2 = 1.5	0.1761	0.1761	0.1761	0	0.1761
in	0	1	1	1	3	3/3 = 1	0	0	0	0	0
of	0	1	1	1	3	3/3 = 1	0	0	0	0	0
silver	1	0	2	0	1	3/1 = 3	0.4771	0.4771	0	0.9542	0
shipment	0	1	0	1	2	3/2 = 1.5	0.1761	0	0.1761	0	0.1761
truck	1	0	1	1	2	3/2 = 1.5	0.1761	0.1761	0	0.1761	0.1761

3.7.4 Length Normalization

Let's consider two documents in the corpus. Also considers, nth page in document D1 and nth and n + 1th pages of the document D2 of similar text. Now, the larger document contains more terms with the more occurrence for a specific term/word. Considering the above scenario, in Document D1, if a term chair appears 7 times and the same term appears 14 times in Document D2, then to overcome this situation length normalization must be applied. One of the length normalization techniques is cosine length normalization.

$$w_{ij} = \frac{w'_{ij}}{\sqrt{\sum_{k=1}^{t} w_{ik}^{t2}}} \tag{3.3}$$

The equation 3.3 is considered for normalizing the weight of the ith term of the document. Here, t indicates the number of terms in the collection of documents, whereas Wij' represents the term frequency without considering normalized length. The Wij represents the length adjusted weight. The term frequency weights are standardized by dividing the length of the document. The equation is given below.

$$Wij = \frac{Wij}{length\ of\ document} \tag{3.4}$$

Here, the length of the document is considered as a relative length of the document.

3.8 Attribute Selection

3.8.1 Introduction

Attribute selection, more popularly known as feature selection is the technique of selecting a subset of relevant features for building robust learning models in machine learning using statistical methods. It is also called variable selection, feature reduction or variable subset selection.

Many attribute / feature selection methods have been developed and extensive research work has already been done in this field.

Feature selection is a process commonly used in machine learning, wherein a subset of the features available from the data is selected for application of a learning algorithm. The best subset contains the least number of dimensions that most contribute to accuracy; we discard the remaining, unimportant dimensions. This is an important stage of preprocessing and is one of two ways of avoiding the curse of dimensionality – the other is feature extraction. It decreases the size of the effective vocabulary and increases accuracy of Text Mining by decreasing noise.

3.8.2 Comparison of Attribute Selection Methods

The most popular attribute selection methods are the frequency distribution, Mutual Information (MI), the chi-square test, correlation coefficient and relevancy score. The methods are all statistical based on probability distributions. The formulas of these methods are given in Table 3.2. The details

about these methods are in the references section as per the reference numbers in the last column of the table.

Table 3-2 Main methods of feature reduction / selection

Function	Denoted by	Mathematical Form
Document Frequency	$\#(t_k, c_i)$	$P(t_k, c_i)$
Information gain (Expected Mutual Information)	$IG = (t_k, c_i)$	$P(t_k, c_i) \cdot \log \frac{P(t_k, c_i)}{P(c_i) \cdot P(t_k)} + P(\overline{t_k}, c_i \cdot \log \frac{P(\overline{t_k}, c_i)}{P(c_i) \cdot P(\overline{t_k})}$
Chi-square	$x^2(t_k, c_i)$	$\frac{g \cdot [P(t_k, c_i) \cdot P(\overline{t_k}, \overline{c_i}) - P(t_k, \overline{c_i}) \cdot P(\overline{t_k}, c_i)]^2}{P(t_k) \cdot P(\overline{t_k}) \cdot P(c_i) \cdot P(\overline{c_i})}$
Correlation coefficient	$CC(t_k, c_i)$	$\frac{\sqrt{g} \cdot [P(t_k, c_i) \cdot P(\overline{t_k}, \overline{c_i}) - P(t_k, \overline{c_i}) \cdot P(\overline{t_k}, c_i)]}{\sqrt{P(t_k) \cdot P(\overline{t_k}) \cdot P(c_i) \cdot P(\overline{c_i})}}$
Relevancy score	$RS(t_k, c_i)$	$\log \frac{P(t_k \mid c_i) + d}{P(\overline{t_k} \mid \overline{c_i}) + d}$

As per the research done by Martin Sewell[1], the different feature selection methods are as follows.

- Kira and Rendell (1992) described a statistical feature selection algorithm called RELIEF that uses instance based learning to assign a relevance weight to each feature.
- John, Kohavi and Pfleger (1994) addressed the problem of irrelevant features and the subset selection problem. They presented definitions for irrelevance and for two degrees of relevance (weak and strong). They also state that features selected should depend not only on the features and the target concept, but also on the induction algorithm. Further, they claim that the filter model approach to subset selection should be replaced with the wrapper model.
- Pudil, Novovi˘cov´a and Kittler (1994) presented “floating” search methods in feature selection. These are sequential search methods characterized by a dynamically changing number of features included or eliminated at each step. They were shown to give very good results and to be computationally more effective than the branch and bound method.
- Koller and Sahami (1996) examined a method for feature subset selection based on Information Theory: they presented a theoretically justified model for optimal feature selection based on using cross-entropy to minimize the amount of predictive information lost during feature elimination.
- Jain and Zongker (1997) considered various feature subset selection algorithms and found that the sequential forward floating selection algorithm, proposed by Pudil, Novovi˘cov´a and Kittler (1994), dominated the other algorithms tested.
- Dash and Liu (1997) gave a survey of feature selection methods for classification. In a comparative study of feature selection methods in statistical learning of text categorization (with a focus is on aggressive dimensionality reduction).
- Yang and Pedersen (1997) evaluated document frequency (DF), information gain (IG), mutual information (MI), a CHI-square test and term strength (TS); and found IG and CHI to be the most effective.

[1]The different feature selection methods as discussed by Martin Sewell, http://www.machine-learning.martinsewell.com/feature-selection

- Blum and Langley (1997) focused on two key issues: the problem of selecting relevant features and the problem of selecting relevant examples.
- Kohavi and John (1997) introduced wrappers for feature subset selection. Their approach searches for an optimal feature subset tailored to a particular learning algorithm and a particular training set.
- Yang and Honavar (1998) used a genetic algorithm for feature subset selection.
- Liu and Motoda (1998) wrote their book on feature selection which offers an overview of the methods developed since the 1970s and provides a general framework in order to examine these methods and categorize them.
- Weston, et al. (2001) introduced a method of feature selection for SVMs which is based upon finding those features which minimize bounds on the leave-one-out error.
- Xing, Jordan and Karp (2001) successfully applied feature selection methods (using a hybrid of filter and wrapper approaches) to a classification problem in molecular biology involving only 72 data points in a 7130 dimensional space.
- Forman (2003) presented an empirical comparison of twelve feature selection methods. Results revealed the surprising performance of a new feature selection metric, 'Bi-Normal Separation' (BNS).

- Guyon and Elisseeff (2003) gave an introduction to variable and feature selection. They recommend using a linear predictor of your choice (e.g. a 2 linear SVM) and select variables in two alternate ways: (1) with a variable ranking method using correlation coefficient or mutual information; (2) with a nested subset selection method performing forward or backward selection or with multiplicative updates.

SUMMARY

This chapter gave a detailed description of the Text Pre-processing tasks which are performed before the Text Summarization model is implemented. Text Pre-processing is in fact very important as the output of the algorithms depend on

how well the text is pre-processed. The pre-processing methods are also interesting areas of research. The next chapter discusses about the Latent Semantic Analysis that is performed on the pre-processed data.