IMPORTANT KEYWORDS EXTRACTION FROM DOCUMENTS USING SEMANTIC ANALYSIS

H. M. MAHEDI HASAN
FALGUNI SANYAL

SUPERVISOR: DR. MD. HAIDER ALI

CO-SUPERVISOR: DIPANKAR CHAKI

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF ENGINEERING & COMPUTER SCIENCE
BRAC UNIVERSITY
MOHAKHALI, DHAKA-1212
BANGLADESH

Important Keywords Extraction from Documents Using Semantic Analysis

A Thesis submitted in
partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science & Engineering of
BRAC University

By

H. M. Mahedi Hasan (13101270)
Falguni Sanyal (13301058)

Supervisor:
Dr. Md. Haider Ali

Co-Supervisor:

Dipankar Chaki

@April 2017

DECLARATION

This is to certify that the research work titled "*Important Keywords Extraction from Documents using Semantic Analysis"* is submitted by H. M. Mahedi Hasan (ID-13101270), and Falguni Sanyal (ID-13301058) to the Department of Computer Science & Engineering, BRAC University in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering. The contents of this thesis have not been submitted elsewhere for the award of any degree or any other publication. We hereby declare that this thesis is our original work based on the results we found. The materials of work found by other researchers and sources are properly acknowledged and mentioned by reference. We carried out our work under the supervision of Dr. Md. Haidar Ali.

Dated: April 10, 2017

Signature of Supervisor

Dr. Md. Haider Ali
Professor
Department of Computer Science & Engineering
School of Engineering & Computer Science
BRAC University

Signature of Co-Supervisor

Dipankar Chaki
Lecturer
Department of Computer Science & Engineering
School of Engineering & Computer Science
BRAC University

Signature of Authors

H. M. Mahedi Hasan (13101270)

Falguni Sanyal (13301058)

BRAC UNIVERSITY

FINAL READING APPROVAL

Thesis Title: Important Keywords Extraction from Documents using Semantic Analysis

Date of Submission: 10th of April, 2017

The final form of the thesis report is read and approved by Dr. Md. Haider Ali. Its format, citations, and bibliographic style are consistent and acceptable. Its illustrative materials including figures, tables, and charts are in place. The final manuscript is satisfactory and is ready for submission to the Department of Computer Science & Engineering, School of Engineering and Computer Science, BRAC University.

Supervisor

_____

Dr. Md. Haider Ali
Professor
Department of Computer Science and Engineering
School of Engineering & Computer Science
BRAC University

ACKNOWLEDGMENT

We are grateful to our thesis supervisor Dr. Md. Haider Ali, Professor, and co-supervisor Dipankar Chaki, Lecturer II, Department of Computer Science & Engineering, for their inspiration, idea, guidance and overall suggestions to improve this work. They have offered us understanding and support at all stages of our work. Their practical suggestions kept us rooted to reality, and helped us to complete our work on time.

We also acknowledge the contribution of S. Mahbub-Uz-Zaman, and Marifa Farzin Reza, former student of BRAC University. They helped us in every step grasping the Natural language processing knowhow for our research work, and provided different data and knowledge required for the thesis work.

Nevertheless, we would like to express our gratitude to Department of Computer Science and Engineering, BRAC University and our teachers for helping us with all the necessary support.

ABSTRACT

Keyword extraction is an automatic selection of terms which describes the content of a document. Keywords define the terms that represent the core information from the documents. In order to go through massive amount of documents to find out the relevant information, keyword extraction will be the key approach. This approach will help us to understand the depth of a document even before we read it. In this research, we have found out different approaches and algorithms that have been used in keyword extraction technique. Conditional random fields (CRF), Support vector machine (SVM), NP-chunk, N-grams, Multiple linear regression, Logistic regression, and semantic analysis has been used to find out important keywords from a document. Immense research shows us that SVM and CRF gives better results where CRF accuracy is greater than SVM based on F1 score (The balance between precision and recall). According to precision, SVM shows better result than CRF. But, in case of recall, logit shows the greater result. Semantic relation between words is also another key feature in keyword extraction techniques. Semantic analysis is very effective field in natural language processing and using semantic relation, it is possible to find out the relation between words as well as between the lines. In this thesis paper, we have used semantic analysis and processing the documents to find out the important keywords from documents.

Index Terms: Natural Language Processing (NLP), Semantic Analysis, TextBlob, NLTK, POS-tagging, N-grams, Keyword, Extraction, Documents

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

NLP:	Natural Language Processing
SVM:	Support Vector Machine
TF:	Term Frequency
ICTCLAS:	Institute of Computing Technology, Chinese Lexical Analysis System
POS-tagging:	Parts of Speech Tagging
NLTK:	Natural Language Toolkit
CRF:	Conditional Random Fields
MLR:	Multiple Linear Regression
Logit:	Logistic Regression
RAKE:	Rapid Automatic Keywords Extraction
KEA:	Keyphrase Extraction Approach
ITF:	Inverse Term Frequency
IDF:	Inverse Document Frequency

CHAPTER 1

1.1 Introduction

Keyword extraction is one of the most important techniques for data analysis. The main task of important keyword extraction is to extract a specific set of words or keywords that can highlight the main content of the document. Automatic indexing, automatic summarization, automatic clustering, automatic filtering, topic detection and tracking, information visualization etc. are the basic data mining applications related to keyword extractions [1]. One of the important methods is statistical approach which is used to identify the keywords based on statistical data. It does not require any training data. Some common approaches are word co-occurrence, PAT-tree, lexical analysis and syntactic analysis which is term frequency and N-grams [2]. Turney (2000) introduced the automatic keyword extraction which is considered as a supervised machine learning task. Also GenEx is proposed by zhang as a machine learning approach and for its execution, vector machine and genetic algorithm have been used. Kupiec, Pedersen, and Chen (1995), Teufel, and Moens (1997) used sentence level features to classify important sentences. Word frequency also used by Kupiec for the classification task. Barzilay and Elhadad (1997) have shown that the features based on lexical chains are good features for text summarization. Turney made a comparison between genetic algorithm and C4.5 decision trees for this task and decided that genetic algorithm provides better results than decision tress. Genetic method solves both constrained and unconstrained optimization problems based on a natural selection process and its main task is to divide a population into individual solutions. C4.5 generates a decision tree to be

used for classification. In [3], conditional random fields have been used to extract keywords in a new probabilistic model for segmenting and labeling sequence data. In the next section, we mentioned different algorithms to be used in the keyword extraction field and then in chapter 3, we discussed our selected approach and key concepts along with that. At the end of our paper, results, discussions, future works and references are given.

1.2 Motivation

In research, a lot of documents, journals, and papers needs to be processed to find out key information. In sentimental analysis fields, keywords defines the core information of the documents. Data analysis requires huge amount of processing in order to identify relative information. At present it is almost impossible to keep track of similar type of documents, journals or research papers altogether. And to process all the words in the documents to find out if they are same or equal importance. Moreover, detecting relevant articles would be slow and time consuming. By extracting keywords from the documents would be a sufficient approach to keep track of similar type of journals, articles or documents. Many algorithm or approach has been introduced to extract keyword from documents. Term frequency, POS-tagging, semantic relation, Support vector machine, C4.5 decision tress, Conditional random fields etc. are already introduced in this field and all approaches has shown better results in extracting keywords but to increase the precision and recall, relation between words and sentences needs to be captured carefully. All these approaches and many possibilities motivates us to work with this topic. We believe that getting better result of extracting keywords from documents is possible using various kinds of methods together and applying it in different fields where lot of data processing is required.

1.3 Literature Review

Keywords can be viewed as a shorten version of documents. Important keywords can be utilize for document retrieval, webpage retrieval, text mining etc. Keyword extraction follows different phases and in [2], Pre-Processing Phase: unification and removing Stopwords has been used and AAS (Attention Attractive Strings) as keywords. Post-Processing phase use different efficient algorithms to find the keywords. In many papers, word occurrence has been considered as one of the most important features. Probability of KWNA (threshold), EPKLN and EPKRN have been used to minimize the threshold of words co-occurrences [2]. KWNA, EPKLN and EPKRN is a probabilistic term which identifies the number of keywords using AAS. AAS is a procedure that compare words to a co-occurrence word. Here, in the topic, eight keywords are extracted from each document as initial list of keywords and then one or two of them are removed using proposed post-processing method and 800 datasets have been used [2]. In keyword, NP-chunks give a better precision than n-grams with adding POS-tagging as an additional feature [4]. The number of words and the frequency of a noun phrases, as well as the frequency of the head nouns is used by Barker and Cornacchia (2000) to determine the keywords from documents. Daille (1994) applied statistical filters on the extracted noun phrases to determine the keywords. In [4], study showed that term frequency is the best filter candidate in keyword extraction technique. N-grams technique removed non-alphanumeric characters that are not keywords in the training dataset. Numbers were removed if they appeared separately and proper nouns were kept. In NP-chunks, nouns contain the content of the documents and manually assigned keywords are happened to be noun or noun phrase with objective. POS-tag patterns shows tagging the word with proper parts of speech such as ADJECTIVE NOUN (singular or mass) NOUN (singular or mass), ADJECTIVE NOUN (plural),

NOUN (plural) etc. From the result, it is shown that N-grams with POS-tag has high F-score [test accuracy] and Chunking with POS-tag has high precision as well as Pattern without POS-tag has high recall [4]. In [5], documents were split into words and processed to find out the noun, verb and phrases as keywords and the process of word splitter is 5 different steps as word splitter initialize, POS-tagging, place and person's name identified, restart word splitter and restart the above steps. Stop words have been removed in the process. ICTCLAS (Institute of Computing Technology, Chinese Lexical Analysis System) uses semantic analysis to improve the accuracy of word splitter. ICTCLAS comprises word dissection, Parts-Of-Speech tagging and unknown words recognition. In keyword extraction technique, authors can set keywords for their documents and those might or might not be occurred in the text. As mentioned before, lexical text chains are effective features for summarization and to make lexical chains, word senses and semantic relations between words should be known [6]. WordNet is used for lexical chain building algorithm and the WordNet can be used for lexical chain builder to synonym, hypernym/hyponym and meronym. Occurrence of words in the documents is key feature in different techniques of important keyword extraction. And it could be viewed as first occurrence in the text, number of occurrence of the words in the text and the last occurrence of the words. C4.5 algorithm used with different features which decreases the variance and increase the accuracy of extracting keywords from the documents [6]. The lexical chain features improve the precision significantly in the keyword extraction process.

1.4 Thesis Outline

Chapter 2

In this chapter we have discussed different effective approaches that show better results in keyword extraction field.

Chapter 3

This chapter reviews natural language processing and python toolkits we chose to develop our keyword extraction procedure.

Chapter 4

It includes the methodology that we used to develop our keyword extraction procedure.

Chapter 5

It contains Semantic analysis and relations along with the discussion of the overview of our selected approach.

Chapter 6

In this chapter, we demonstrate the implementation phase of our approach.

Chapter 7

In this chapter, we displayed the implementation results and comparison with other approaches.

Chapter 8

We discussed conclusion along with future work.

CHAPTER 2

2.1 Conditional Random Fields:

Sentence segmenting and labelling is an application of conditional random fields (CRF). It is used in keywords extraction that showed better result than the other approaches. The main approaches of this algorithm is, preprocessing and features extraction, CRF model training, and CRF labeling and keyword extraction, results evaluation [3]. In [3], CRF++ tool with POS-tag is used to extract keywords from the documents.

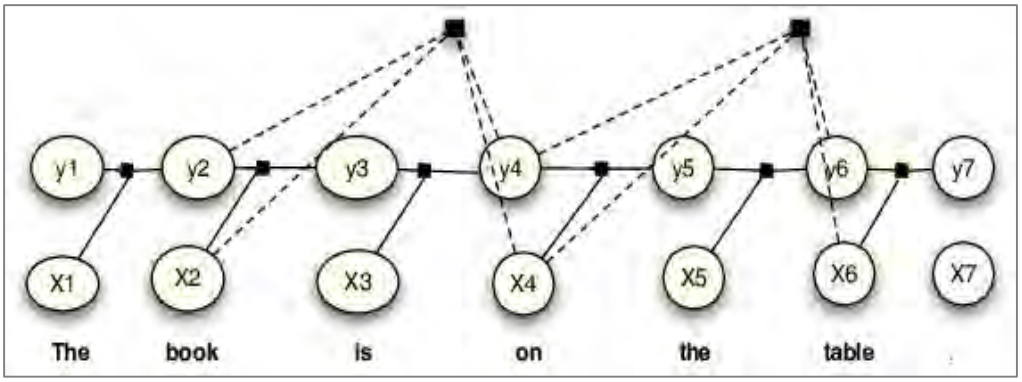

Figure 2.1 Graphical representation of CRF with preposition template. Prepositions are connected to the candidate trajectors and candidate landmarks [noun phrases]. Factors occur as black squares.

2.2 SVM, MLR and Logit:

Support vector machine (SVM), multiple linear regression (MLR), logistic regression (logit) has been used in this approach along with TF*IDF (Term frequency and inverse document frequency) as baseline to extract keyword from the documents [3]. SVM is supervised learning process which categorizes other inputs and produce an optimal output. And the result shows that SVM and CRF gives better results where CRF accuracy is greater than SVM based on F1 score. According to precision, SVM shows better result than CRF. But, in case of recall, logit shows the greater result. Logistic regression model displays better result than multiple linear regression model in the task of keyword extraction [3]. As mentioned for future works, conditional random fields and semantic relation between keywords to perform on the greater number of texts.

2.3 Statistics, Machine and Semantic Learning:

Keyword or Keyphrase extraction could be categorize into three category: statistics, machine learning, and semantically matching. P. Turney developed Extractor and GenEx which is machine learning based extraction. TRUCKS is the keyphrase extraction technique between statistical and sentimental approaches which shows better results in extracting keyphrases from the documents but also reject actual keyphrases [8]. Jordi Vivaldi et.al [8] proposed AdaBoost Algorithm to find higher accuracy of keyphrase extraction system which has also some disadvantages and it categorize into two groups, one is linguistic and another statistics. In [8], main functions were split into two group, training mode and recovery mode. Training mode initialize the main knowledge base and recovery mode recovers the false rejected key phrases thus being reevaluated to find out keyphrases from the documents. Statistical methods uses word frequency, term frequency, and word co-occurrences which provides some good results. In [10], linguistic features increase

the performance of extracting keywords from the documents. Rapid automatic keywords extraction (RAKE) used to extract keywords to include a lists of StopWords, phrase delimiters, and word delimiters [10]. In RAKE, stop words and phrase delimiters have been used to divide the document text into candidate keywords and "stemmer function" which converts all the plural words to singular words, and removing suffixes based on different features. Adaptive lesk algorithm model will calculate the word-to-word similarity for all word pairs [11]. Statistics-based methods such as Bayesian, K-Nearest Neighbor, and Expectation-Maximization. As mentioned before in both statistical and machine learning approach, TF (Term frequency), ITF (Inverse Term Frequency), IDF (Inverse Document Frequency), position of word in document, position in paragraphs, position in sentences, modified TF, occurrence in first and last paragraphs, occurrence in abstract, headings, figures, and tables have been used. In Keyword extraction techniques, these approaches provide an essential impact on huge data set analysis, text analysis, and document summarization etc. fields.

2.4 Keyphrase Extraction Approach:

KEA (keyphrase extraction approach) used a machine learning methodology based on naïve Bayes decision rule [9]. TF-IDF scores can be used to differentiate between Keyphrase and non-Keyphrase. In [9], noun factors are determined by its frequency in the document, its composition and how specific these words and sub-phrases are in the domain of the document predefined as domain specific keywords or keyphrases in the database.

Keywords extraction methodologies can be categorize into two features, quantitative and qualitative. Qualitative techniques based on semantic relation [12]. Semantic relatedness supports many relations such as Hypernym, Hyponym, Holonym, Meronym, Troponym, and Antonym. All these approaches have been applied in this field and get results under different methodologies.

CHAPTER 3

3.1 Natural Language Processing

Interactions between human language and computers is called Natural Language Processing. NLP is a way for computers to analyze, understand, and derive meaning from human language in a smart and efficient way. By utilizing NLP, developers can organize and structure knowledge to perform tasks such as automatic summarization, translation, named entity recognition, relationship extraction, sentiment analysis, speech recognition, and topic segmentation.

NLP has open source libraries and some of them are mentioned below:

- **Apache OpenNLP:** a machine learning toolkit that provides tokenizers, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, the coreference resolution system and more.

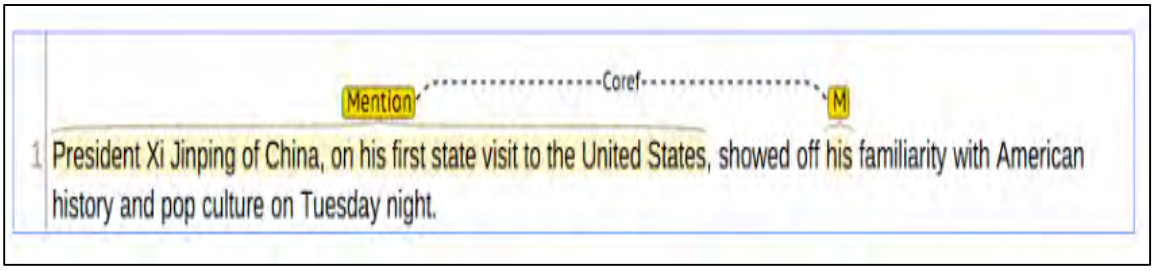

Figure 3.1 Coreference

- **Natural Language Toolkit (NLTK):** A Python library that provides modules for processing text, classifying, tokenizing, stemming, tagging, parsing, and more.

- **Stanford CoreNLP:** A suite of NLP tools that provide part-of-speech tagging, the named entity recognizer, sentiment analysis, and more.

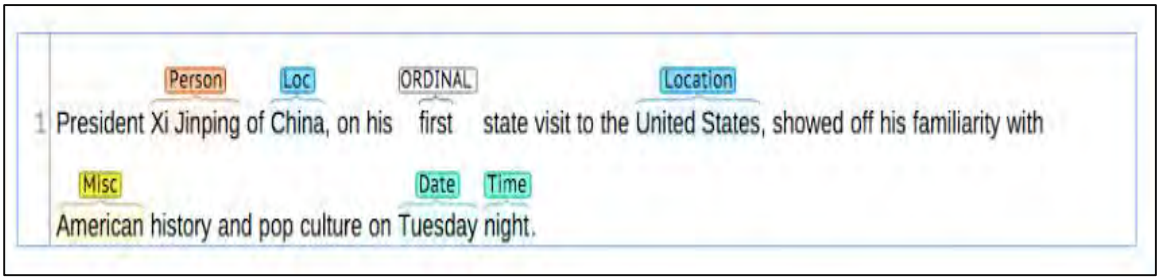

Figure 3.2 Name Entity Recognition

Some Application of NLP has given below:

- Automatically summarize a document, exacting topic sentences from text analysis.
- Extracting keyword topic tags from a document using LDA (Latent Dirichlet Allocation), which determines the most relevant words from a document.
- Sentiment analysis etc.

3.2 Natural Language Toolkit:

NLTK is a leading platform for building Python programs to work with human language data. It provides easy interfaces over 50 corpora and lexical resources such as WordNet and text processing libraries for classification, tokenization, stemming, tagging, parsing and semantic reasoning etc. In this research, we used different modules from NLTK toolkit.

NLTK libraries need to be imported in order to use its different modules. Some of them are described below:

- **Tokenize and POS-Tagging**

  In that process, texts are being processed and tokenized to separate the words and then tag the word with their corresponding pasts of speech.

```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning
... Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning',
'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
>>> tagged = nltk.pos_tag(tokens)
>>> tagged[0:6]
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'),
('Thursday', 'NNP'), ('morning', 'NN')]
```

Figure 3.3 Example of Tokenize and POS-tagging

- **Identification of Name Entities**

  By using NLTK toolkit, words in the text being tagged by NLTK method and use nltk.chunk.ne_chunk(tagged) method to find out the named entities.

```
>>> entities = nltk.chunk.ne_chunk(tagged)
>>> entities
Tree('S', [('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'),
          ('on', 'IN'), ('Thursday', 'NNP'), ('morning', 'NN'),
      Tree('PERSON', [('Arthur', 'NNP')]),
          ('did', 'VBD'), ("n't", 'RB'), ('feel', 'VB'),
          ('very', 'RB'), ('good', 'JJ'), ('.', '.')])
```

Figure 3.4 Named Entity Identification

❖ Parse Tree

Using treebank of corpus interface from NLTK toolkit, a text file can be represented as a tree after being processed by POS-tagging.

```
>>> from nltk.corpus import treebank
>>> t = treebank.parsed_sents('wsj_0001.mrg')[0]
>>> t.draw()
```

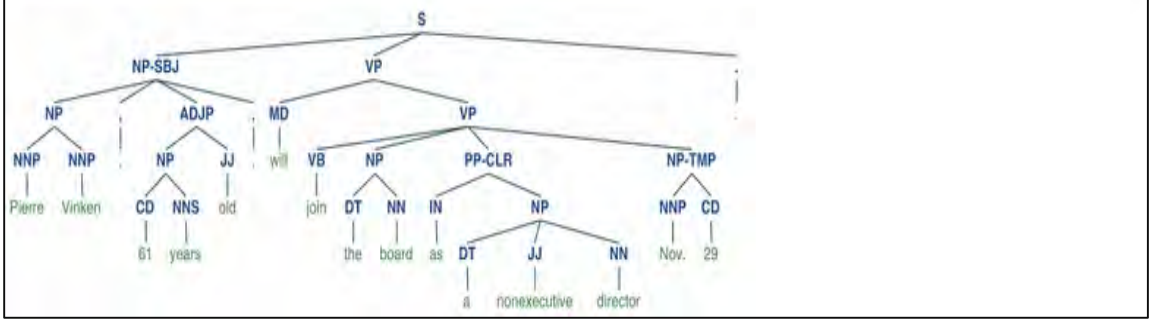

Figure 3.5 Parse Tree

3.3 TextBlob: Text Processing

TextBlob is a python library for processing textual data. It provides a simple API for simple natural language processing tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more. This is similar to NLTK toolkit and provides the same and some more advanced modules as well. In this research, we use N-grams and sentence making modules from TextBlob.

Some modules from TextBlob are given below:

- **Create TextBlob and Noun Phrase Extraction**

  To create TextBlob, textblob must be imported and using noun_phrase module, noun phrases could be found from a text file.

```
from textblob import TextBlob
Wiki = TextBlob("Python is a high-level, general-purpose programming language.")
 wiki.tags
[('Python', 'NNP'), ('is', 'VBZ'), ('a', 'DT'), ('high-level', 'JJ'), ('general-purpose', 'JJ'), ('programming', 'NN'), ('language', 'NN')]
wiki.noun_phrases
WordList(['python'])
```

Figure 3.6 Noun Phrase Extraction

- **Wordlists**

  A WordList is just a Python list with additional methods.

```
>>> animals = TextBlob("cat dog octopus")
>>> animals.words
WordList(['cat', 'dog', 'octopus'])
>>> animals.words.pluralize()
WordList(['cats', 'dogs', 'octopodes'])
```

Figure 3.7 Wordlists and pluralize

❖ **Word Frequency**

In order to find a word frequency from a text, word_coutns module could be used and it will return the number of occurrence of the specific word.

```
>>> monty = TextBlob("We are no longer the Knights who say Ni. "
...                  "We are now the Knights who say Ekki ekki ekki PTANG.")
>>> monty.word_counts['ekki']
3
```

Figure 3.8 Word Frequency Calculation

❖ **Sentiment Analysis**

The sentiment property returns a namedtuple of the form Sentiment (polarity, subjectivity). The polarity score is a float within the range [-1.0, 1.0]. The subjectivity is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective.

```
>>> testimonial = TextBlob("Textblob is amazingly simple to use. What great fun!")
>>> testimonial.sentiment
Sentiment(polarity=0.39166666666666666, subjectivity=0.4357142857142857)
>>> testimonial.sentiment.polarity
0.39166666666666666
```

Figure 3.9 Sentiment Analysis of text

CHAPTER 4

4.1 Term Frequency

Term frequency is calculating the occurrence of words in the text file. After processing the documents, we detect the term frequency of every words that plays a big role in keyword extraction technique. In this thesis, we used NLTK toolkit for calculating term frequency of every words.

Term frequency of a document has shown below:

Document:

“Harry Potter is a series of fantasy novels written by British author J K Rowling. The novels chronicle the life of a young wizard, Harry Potter, and his friends Hermione Granger and Ron Weasley, all of whom are students at Hogwarts School of Witchcraft and Wizardry. The main story arc concerns Harry's struggle against Lord Voldemort, a dark wizard who intends to become immortal, overthrow the wizard governing body known as the Ministry of Magic, and subjugate all wizards and Muggle.”

Coding Implementation

```
#Frequency Calculating
c = 0
GetText = open('Random.txt', 'r')
MakeText = open('krypton.txt', 'w')
for word in GetText:
    for SubWord in word.split():
        SubWord = SubWord.lower()
        if SubWord in StopWord:
            continue
        else:
            Find = re.findall('(.+)\.', SubWord)
            Find1 = re.findall('(.+)\,',SubWord)
            if len(Find) > 0:
                SubWord = Find[0]
                c = 1
            if len(Find1) > 0:
                SubWord = Find1[0]
            MakeText.write(SubWord + " ")
            Database[SubWord] = Database.get(SubWord,0) + 1  # Frequency Calculating
            if c == 1 :
                c = 0
                MakeText.write("\n")


GetText.close()
MakeText.close()
```

Figure 4.1 Term Frequency Calculating Implementation

Table 4.1 Term Frequency Calculation

Words	Frequency
novels	2
wizard	3
Potter	2
friends	1
magic	1
dark	1
immortal	1

4.2 Stemming

In the documents, every words needs to be processed to find out the keyword. Because of the words formation, the system could miss out same word. By stemming the words which means removing the suffixes and plural to singular words, the system could determine the same word and their relation among the sentences and occurrence in the documents.

Stemming words from a document has shown below:

Document:

“Harry Potter is a series of fantasy novels written by British author J K Rowling. The novels chronicle the life of a young wizard, Harry Potter, and his friends Hermione Granger and Ron Weasley, all of whom are students at Hogwarts School of Witchcraft and Wizardry. The main story arc concerns Harry's struggle against Lord Voldemort, a dark wizard who intends to become immortal, overthrow the wizard governing body known as the Ministry of Magic, and subjugate all wizards and Muggle.”

Coding Implementation

```
#Stemming with Snowball
Snow_Stemmer = SnowballStemmer("english")
SnowStem = open('SnowballStem.txt', 'w')
rd = open('krypton.txt', 'r')

SnowD=0
for word in rd:
    if SnowD == 1:
        SnowStem.write("\n")
    SnowD = 1
    for SubWord in word.split() :
        take = Snow_Stemmer.stem(SubWord)
        SnowStem.write(take + " ")

SnowStem.close()
rd.close()
```

Figure 4.2 Stemming

Table 4.2 Stemming of words in Documents

Before Stemming	After Stemming
novels	novel
friends	friend
fantasy	fantasi
intends	intend
concerns	concern
wizards	wizard
students	student

4.3 Parts of Speech Tagging

Understanding the words relation between sentences is important. Mostly nouns or adjective contains key information of the documents. In order to efficiently analyze of the words, the system will POS-tag of every words in the documents which it will define the words parts of speech.

In this thesis, we used NLTK toolkit for POS-tagging in the documents.

Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year; home, costs, time, Africa*
NUM	numeral	*twenty-four; fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per; that, up, with*
PRON	pronoun	*he, their; her; its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

Figure 4.3 Universal Part-of-Speech Tagset

POS-tagging words from a document has shown below:

Document

“Harry Potter is a series of fantasy novels written by British author J K Rowling. The novels chronicle the life of a young wizard, Harry Potter, and his friends Hermione Granger and Ron Weasley, all of whom are students at Hogwarts School of Witchcraft and Wizardry. The main story arc concerns Harry's struggle against Lord Voldemort, a dark wizard who intends to become immortal, overthrow the wizard governing body known as the Ministry of Magic, and subjugate all wizards and Muggle.”

Code Implementation

```
# Start poss tagging the words
rd = open('krypton.txt', 'r')
PosT = open('PosTag.txt','w')

tokenize_to_posTag = PunktSentenceTokenizer(rd)

for word in rd:
    words = word_tokenize(word)
    tagged = nltk.pos_tag(words)
    print(tagged)
    for value in tagged:
        PosT.write(value[0]+" "+value[1])
        PosT.write("\n")
        postag[value[0]] = value[1]
PosT.close()
rd.close()
```

Figure 4.4 POS-tagging

Table 4.3 POS-tagging words in Documents

Words	POS-tagging
harry	NN
written	VBN
young	JJ
concerns	NNS
chronicle	VBP
british	JJ
life	NN

4.4 N-grams

N-grams of texts are being used frequently in text mining and natural language processing tasks. It is basically a set of co-occurring words within a given document and when computing N-grams, we move one or more words forward. It depends on the number given in the N-gram calculation. N-grams of a sentence has shown below:

Coding Implementation

```
from textblob import TextBlob
blob = TextBlob("Now is better than never.")
print (blob.ngrams(n=2))
```

Figure 4.5 N-grams [1]

Result of N-grams

```
[WordList(['Now', 'is']), WordList(['is', 'better']), WordList(['better', 'than']), WordList(['than', 'never'])]
```

Figure 4.5 N-grams[2]

4.5 StopWords

Documents contains grammatical words and other supporting words or POS to complete the sentences which doesn't contains any core information of the documents. In order to increase the processing speed, the system will remove the StopWords from the documents at the start of the procedure. In the selected topic, we collected a collection of stop words which contains almost 300 stop words. During the process of a document, stop words has been removed for further implementation procedure.

Removing stop words has shown below:

Document

"Harry Potter is a series of fantasy novels written by British author J K Rowling. The novels chronicle the life of a young wizard, Harry Potter, and his friends Hermione Granger and Ron Weasley, all of whom are students at Hogwarts School of Witchcraft and Wizardry. The main story arc concerns Harry's struggle against Lord Voldemort, a dark wizard who intends to become immortal, overthrow the wizard governing body known as the Ministry of Magic, and subjugate all wizards and Muggle."

Stop Words

['is', 'a', 'of', 'by', 'the', 'and', 'his', 'all', 'are', 'at', 'main', 'to', 'as']

CHAPTER 5

5.1 Semantic Analysis

Semantic relatedness can be viewed as many relations such as Hypernym, Hyponym, Holonym, Meronym, Troponym, and Antonym. It is shows the relation between words or meaning of words at word level as well as meaning of sentences and keyphrases at sentence level. Using semantic relation between words will be a key feature in the system because it can provide meaning to a word that will help to identify the keywords from the documents.

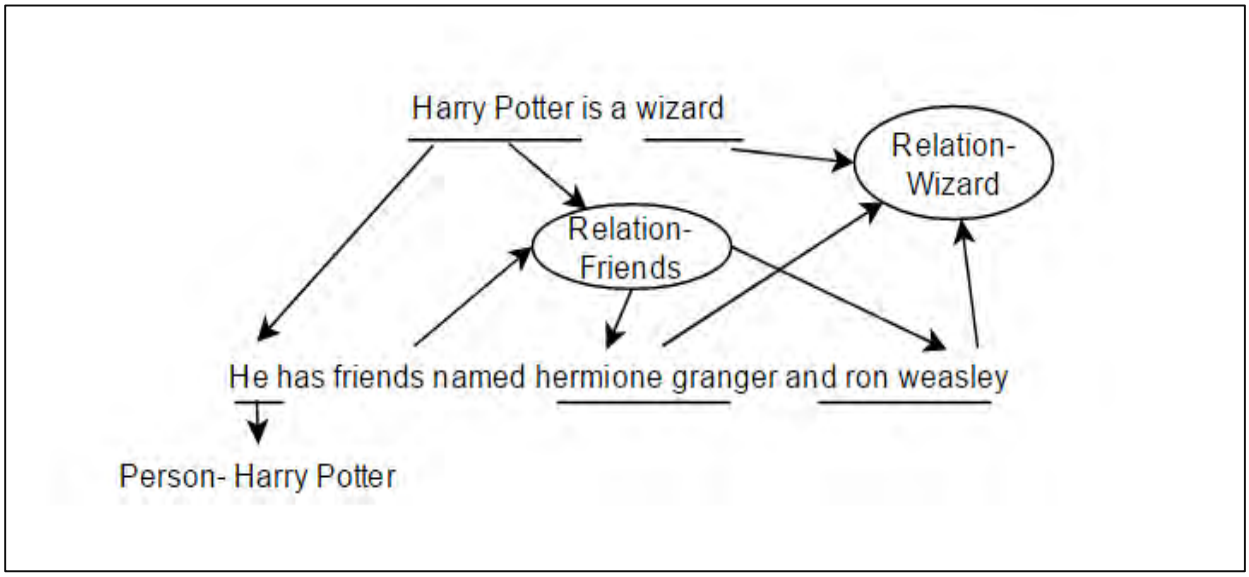

Figure 5.1 Semantic Relation

Semantic relation:

In figure 5.1, there is two relation [relation friends] and [relation wizard] mentioned between Harry Potter, Hermione Granger and Ron Weasley. In the second sentence, he means the Harry Potter and friends word connect him with Hermione and Ron. In that case Harry is a wizard and so his friends will be wizards too. In semantic analysis, words to words or sentence to sentence semantic relation could be found. By using semantic relation, important keyword could be detected from the documents.

Table 5.1 Semantic relation table of figure 5.1

Words	Relations
Harry Potter	Wizard
Harry Potter	He
Harry Potter	Friends
Friends	Hermione and Ron
Hermione and Ron	Wizard

By using semantic relation, the important keywords would be “Wizard” and “Friends” which makes the relation between the Nouns [Figure 5.1]

5.2 Overview of Our Approach:

By doing immense research, we have found out different approach in natural language field related to keywords extraction technique. From various methods, semantic relation caught our eyes and we developed a complete different approach to extract keywords from the documents. In our approach, we process 500 documents and remove the StopWords at first stage. Then POS-tagging, N-grams, calculating the term frequency of every words from the documents. We separate the nouns and select half of them according to their highest term frequency and made nouns to nouns relation which means if there is more than one noun in a sentence, they are connected to each other. And made nouns to words relation which denotes except noun, rest of the words will be placed under the nouns that appears in the same sentence. We also did the stemming in order to select the keywords and separate the WH questions sentence to give them higher priority. In WH question sentence, it denotes something important that relates to the whole documents. We selected 8 keywords from the documents along with N-grams which if there is a continuous words sequence then it will be count as a keyword by using N-grams (2).

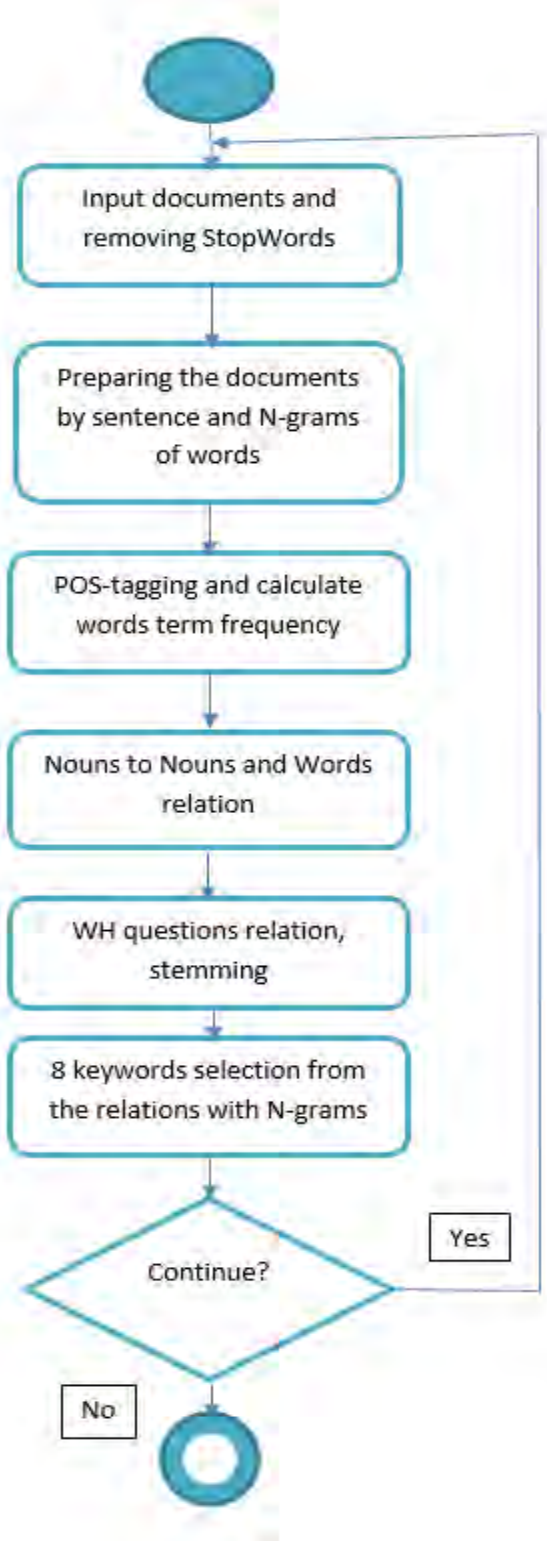

Figure 5.2 Flowchart of our approach

CHAPTER 6

6.1 Documents

We collected 500 documents from different Books and Journal papers to continue our research works.

Sample Document:

The performance of automatic keyword extraction methods is evaluated by recall and precision criteria. The result of keyword extraction usually improves when the selected keywords get closer to the ones suggested by a person. Since recall and precision has mutual effect on each other, increase in precision leads to recall decrease and vice versa. In this paper, a post-processing phase based on using attention attractive strings is proposed, which evaluates candidate keywords acquired from other keyword extraction methods to find the ones closest to the human point of view. In other words, an innovative keyword evaluation function, which is inspired by total probability theorem, is employed to improve the precision criterion in a Farsi automatic keyword extraction task. The attention attractive strings are selected by a reverse-engineering process from 800 Farsi keywordassigned documents. Results indicate that apart from extraction method, improving precision criterion is possible using the proposed post-processing phase without noticeable decrement in recall criterion.

Figure 6.1 Document-1

```
Breadth-first search is a simple strategy in which the root node is expanded first, then all the
SEARCH
successors of the root node are expanded next, then their successors, and so on. In general,
all the nodes are expanded at a given depth in the search tree before any nodes at the next
level are expanded.
Breadth-first search is an instance of the general graph-search algorithm in
which the shallowest unexpanded node is chosen for expansion. This is achieved very simply
by using a FIFO queue for the frontier. Thus, new nodes (which are always deeper than their
parents) go to the back of the queue, and old nodes, which are shallower than the new nodes,
get expanded first. There is one slight tweak on the general graph-search algorithm, which is
that the goal test is applied to each node when it is generated rather than when it is selected for
expansion. This decision is explained below, where we discuss time complexity. Note also
that the algorithm, following the general template for graph search, discards any new path to
a state already in the frontier or explored set; it is easy to see that any such path must be at
least as deep as the one already found. Thus, breadth-first search always has the shallowest
path to every node on the frontier.
```

Figure 6.2 Documents-2

6.2 StopWords

After processing the documents, we remove the StopWords form every document. We collected the StopWords from various sources.

```
a about above above across after afterwards again against all ake almost alone along already also although always am among amongst
amount an and another any anyhow anyone anything anyway anywhere are around as at back be became because become becomes
becoming been before beforehand behind being below beside besides between best beyond bill both bottom but by bye call can co
con could couldn't cry de describe detail do done down due during each e.g. eight either eleven else elsewhere empty
enough etc. even ever every everyone everything everywhere except few fifteen fifty fill find fire first five for former formerly
forty found four from front full further get give go had has hasn't have he hence her here hereafter hereby
herein hereupon hers herself him himself his how however hundred i i.e. if in indeed interest into is it its
itself keep last latter latterly least less ltd madam made mam many may me meanwhile might mill mine more moreover most mostly
move much must my myself name namely neither never nevertheless next nine no nobody none no one nor nothing now nowhere
of off often on once one only onto or other others otherwise our ours ourselves out over own part per
perhaps please put rather re same see seem seemed seeming seems serious several she should show side since sincere sir
six sixty so some somehow someone something sometime sometimes somewhere still such system take ten than that the their them
themselves then thence there thereafter thereby therefore therein thereupon these they thick thin third this those though three through throughout
thru thus to together too top toward towards twelve twenty two un under until up upon us very via was way
we were whatever whenever after whereas whereby wherein whereupon wherever whether while whither will with within without yet you your
yours yourself yourselves & the not use
```

Figure 6.3 StopWords

6.3 N-grams:

After removing the StopWords from the sentence and separate the sentence by processing the documents, we use the N-grams to make phrase with the size 2. We use TextBlob to implement the N-grams modules.

Coding Implementation:

```
rd1 = open((MakeTxt + ".txt"), 'r')
   #N-grams

   for word in rd1:
       blob = TextBlob(word)
       gram = blob.ngrams(n=2)
       for i, j in gram:
           #print (i+" "+j)
           N_grams[i]=j
           #N_grams[j]=i
   rd1.close()
```

Figure 6.4 N-grams Implementation

Result:

This is the result of [Figure 5.3 Document-1]

```
{'methods': 'ones', 'proposed': 'postprocessing', 'point': 'view', 'acquired': 'keyword', 'results': 'indicate',
'leads': 'recall', 'improving': 'precision','paper': 'postprocessing', 'decrease': 'vice', 'result': 'keyword',
'human': 'point', 'innovative': 'keyword', 'keywords': 'acquired', 'farsi': 'keywordassigned',
'total': 'probability', 'decrement': 'recall', 'reverseengineering': 'process', 'closer': 'ones',
'mutual': 'effect', 'based': 'using', 'probability': 'theorem', 'candidate': 'keywords', 'indicate': 'apart',
'selected': 'reverseengineering', 'when': 'selected', 'increase': 'precision', 'extraction': 'method', 'other': 'increase',
'criterion': 'possible', 'attractive': 'strings', 'which': 'inspired', 'theorem': 'employed', 'keywordassigned': 'documents',
'performance': 'automatic', 'apart': 'extraction', 'evaluation': 'function', 'method': 'improving', 'possible': 'using',
'function': 'which', 'process': 'farsi', 'closest': 'human', 'attention': 'attractive', 'precision': 'criterion',
'effect': 'other', 'ones': 'closest', 'words': 'innovative', 'employed': 'improve', 'phase': 'noticeable', 'using': 'proposed',
'improve': 'precision', 'suggested': 'person', 'noticeable': 'decrement', 'keyword': 'extraction', 'vice': 'versa', 'recall': 'criterion',
'inspired': 'total', 'evaluated': 'recall', 'improves': 'when', 'postprocessing': 'phase',
'usually': 'improves', 'evaluates': 'candidate', 'automatic': 'keyword', 'strings': 'selected'}
```

Figure 6.5 Result of N-gram(n==2)

6.4 POS-tagging:

In the next task, we have done POS-tagging and separate Nouns from the document to continue our implementation.

Coding Implementation:

```
## POS Tagging

   rd = open((MakeTxt + ".txt"), 'r')
   PosT = open((PosTag + ".txt"), 'w')

   tokenize_to_posTag = PunktSentenceTokenizer(rd)
   #print("Start Pos Tagging!!: ")


   for word in rd:
       words = word_tokenize(word)
       tagged = nltk.pos_tag(words)
       #print(tagged)
       for value in tagged:
           #print(value[0]+" "+value[1])
           if value[0] in nsw:
               continue
           else:
               if value[0] in StopWord:
                   continue
               else:
                   All_Tagged[value[0]]=value[1]
                   PosT.write(value[0] + " " + value[1])
                   PosT.write("\n")
                   if(value[1]=='NN'):
                       # Frequency Calculating
                       NounsTF[value[0]] = NounsTF.get(value[0], 0) + 1
                       if value[0] in nouns:
                           continue
                       else:
                           nouns.append(value[0])


   PosT.close()
   rd.close()
```

Figure 6.6 POS-tagging implementation

Result:

This is the result of [Figure 5.3 Document-1]


```
{'methods': 'NNS', 'proposed': 'VBN', 'point': 'NN', 'acquired': 'VBD', 'results': 'NNS', 'leads': 'VBZ',
 'improving': 'VBG', 'paper': 'NN', 'decrease': 'NN', 'result': 'NN', 'human': 'JJ', 'innovative': 'JJ',
 'keywords': 'NNS', 'farsi': 'NN', 'total': 'JJ', 'decrement': 'NN', 'reverseengineering': 'VBG',
 'closer': 'RBR', 'mutual': 'JJ', 'documents': 'NNS', 'based': 'VBN', 'probability': 'NN', 'candidate': 'NN',
 'indicate': 'VBP', 'theorem': 'NN', 'selected': 'VBD', 'when': 'WRB', 'increase': 'NN', 'extraction': 'NN',
 'criterion': 'NN', 'attractive': 'JJ', 'versa': 'NN', 'criteria': 'NNS', 'keywordassigned': 'VBD',
 'performance': 'NN', 'apart': 'JJ', 'evaluation': 'NN', 'method': 'NN', 'possible': 'JJ', 'function': 'NN',
 'process': 'NN', 'which': 'WDT', 'closest': 'VBP', 'attention': 'NN', 'precision': 'NN', 'effect': 'NN',
 'ones': 'NNS', 'words': 'NNS', 'employed': 'VBN', 'phase': 'NN', 'using': 'VBG', 'improve': 'VB', 'suggested': 'VBD',
 'task': 'NN', 'noticeable': 'JJ', 'keyword': 'NN', 'vice': 'NN', 'recall': 'NN', 'inspired': 'VBD', 'evaluated': 'VBD',
 'person': 'NN', 'improves': 'VBZ', 'postprocessing': 'VBG', 'usually': 'RB', 'evaluates': 'VBZ',
 'automatic': 'JJ', 'strings': 'NNS', 'view': 'NN'}
```


Figure 6.7 Result of POS-tagging

6.5 Term Frequency

In the next step, we calculated the term frequency of every words from the documents and used it as a key feature in our thesis work.

Code Implementation:

```
##Term Frequency Calculation
    rd = open((MakeTxt + ".txt"), 'r')
    PosT = open((PosTag + ".txt"), 'w')

    tokenize_to_posTag = PunktSentenceTokenizer(rd)
    #print("Start Pos Tagging!!: ")


    for word in rd:
        words = word_tokenize(word)
        tagged = nltk.pos_tag(words)
        #print(tagged)
        for value in tagged:
            #print(value[0]+" "+value[1])
            if value[0] in nsw:
                continue
            else:
                if value[0] in StopWord:
                    continue
                else:
                    All_Tagged[value[0]]=value[1]
                    PosT.write(value[0] + " " + value[1])
                    PosT.write("\n")
                    Allwords[value[0]] = Allwords.get(value[0], 0) + 1
                    if(value[1]=='NN'):
                        # Frequency Calculating
                        NounsTF[value[0]] = NounsTF.get(value[0], 0) + 1
                        if value[0] in nouns:
                            continue
                        else:
                            nouns.append(value[0])


    PosT.close()
    rd.close()
```

Figure 6.8 Term frequency Implementation

Result:

This is the result of [Figure 5.3 Document-1]

{'methods': 2, 'proposed': 2, 'point': 1, 'acquired': 1, 'results': 1, 'leads': 1, 'improving': 1, 'paper': 1, 'decrease': 1,
'result': 1, 'human': 1, 'innovative': 1, 'keywords': 2, 'farsi': 2, 'total': 1, 'decrement': 1, 'reverseengineering': 1, 'closer': 1,
'mutual': 1, 'documents': 1, 'based': 1, 'probability': 1, 'candidate': 1, 'indicate': 1, 'theorem': 1, 'selected': 2, 'when': 1, 'increase': 1,
'extraction': 5, 'criterion': 3, 'attractive': 2, 'versa': 1, 'criteria': 1, 'keywordassigned': 1, 'performance': 1, 'apart': 1, 'evaluation': 1,
'method': 1, 'possible': 1, 'function': 1, 'process': 1, 'which': 2, 'closest': 1, 'attention': 2, 'precision': 5, 'effect': 1, 'ones': 2, 'words': 1,
'employed': 1, 'phase': 2, 'using': 2, 'improve': 1, 'suggested': 1, 'task': 1, 'noticeable': 1, 'keyword': 5, 'vice': 1, 'recall': 4, 'inspired': 1,
'evaluated': 1, 'person': 1, 'improves': 1, 'postprocessing': 2, 'usually': 1, 'evaluates': 1, 'automatic': 2, 'strings': 2, 'view': 1}

Figure 6.9 Result of term frequency

6.6 Stemming:

Some keywords could be missed out because of plurals and –ing postfix and so on. In order to avoid the circumstances, we also implement the stemming module using NLTK.

Coding Implementation:

```
#stemming
    stem = list()
    Snow_Stemmer = SnowballStemmer("english")
    for W in MainAlgo:
        take = Snow_Stemmer.stem(W)
        stem.append(take)


    print stem
```

Figure 6.10 Stemming Implementation

Result:

This is the result of Nouns from [Figure 5.3 Document-1]

```
['task', 'keyword', 'vice', 'process', 'recal', 'resword', 'attent',
'precis', 'extract', 'criterion', 'result', 'versa', 'theorem', 'phase', 'view']
```

Figure 6.11 Result of stemming

6.7 Nouns to Nouns and Words Relation

In our thesis, we create a different approach based on semantic relation between words and sentences. We select the nouns from the documents and take them as heart of the sentence and crate the relation between nouns to nouns and words. Using the relation, we extract 8 keywords from the documents using TF.

Graphical representation of Nouns to Nouns and Words relation:

This graph is based on the [Figure 6.1 Document-1]

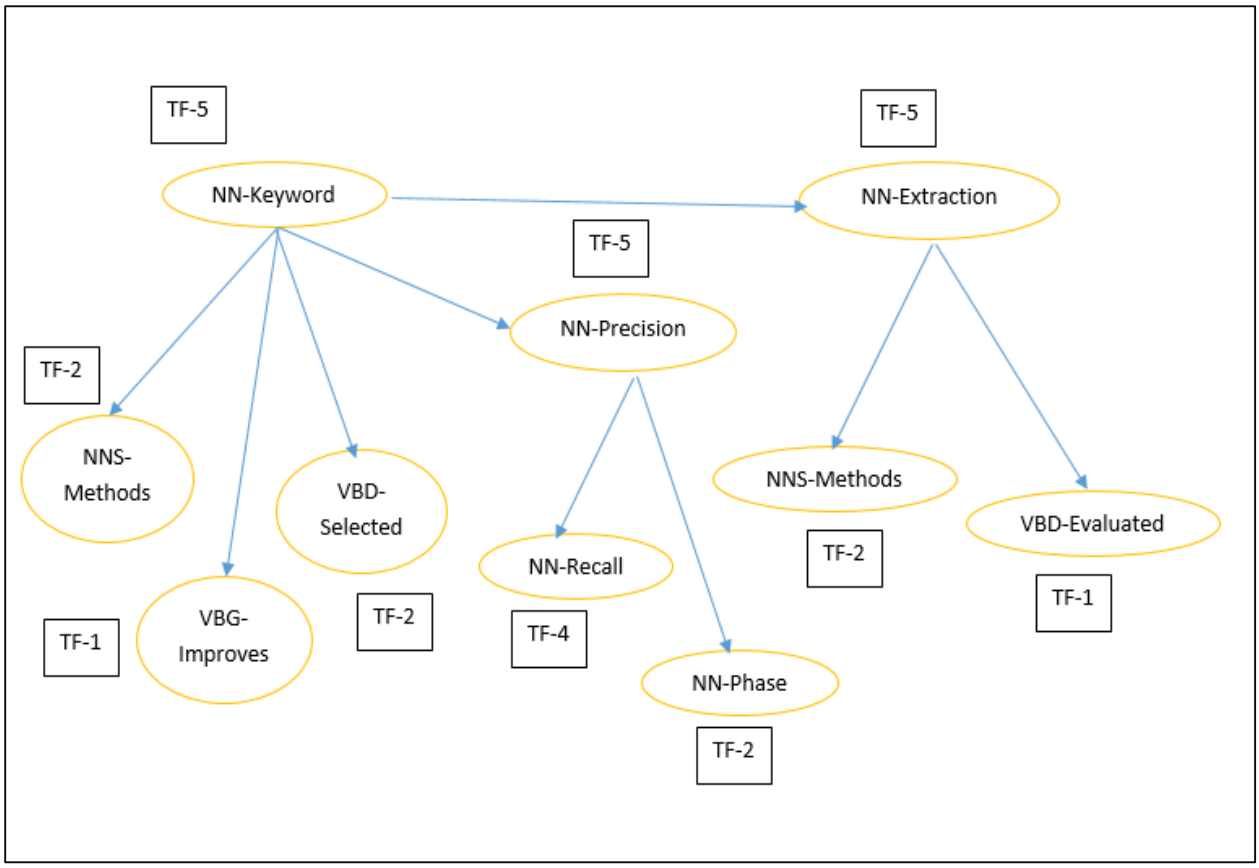

Figure 6.12 Graphical Representation of relations

Coding Implementation:

```
#Relation making Nouns to Nouns and Words
   rd2 = open((MakeTxt + ".txt"), 'r')
   for word in rd2:
       for subword in word.split():
           PickWord = subword
           if PickWord in selectednouns:
               if PickWord in pickNouns:
                   continue
               else:
                   if len(pickNouns) != 0:
                       for fword in pickNouns:
                           if PickWord in MainAlgoRelation[fword]:
                               continue
                           else:
                               if PickWord in StopWord:
                                   continue
                               else:
                                   MainAlgoRelation[fword].append(PickWord)
                       pickNouns.append(PickWord)
                   else:
                       pickNouns.append(PickWord)
           else:
               if len(pickNouns) == 0:
                   if PickWord in selectednouns:
                       continue
                   else:
                       take = Snow_Stemmer.stem(PickWord)
                       if take in selectednouns:
                           continue
                       else:
                           if PickWord in Important:
                               MainAlgo['resWord'].append('PH')
                           else:
                               MainAlgo['resWord'].append(PickWord)
               else:
                   for tok in pickNouns:
                       if PickWord in StopWord:
                           continue
                       else:
                           if PickWord in Important:
                               MainAlgo[tok].append('PH')
                           else:
                               MainAlgo[tok].append(PickWord)

       del pickNouns[:]
   rd2.close()
```

Figure 6.13 Relations Implementation

6.8 WH relations

If there is any WH questions in the documents then it will get higher priority and have a high priority section to select keywords from the documents using TF.

Coding Implementation:

```
questions = list()
for ik in MainAlgo:
    for get in MainAlgo[ik]:
        if get == 'PH':
            #print get
            questions.append(ik)
            for get1 in MainAlgo[ik]:
                if get1 == 'PH':
                    continue
                else:
                    questions.append(get1)
            break

print questions
```

Figure 6.14 WH relation Implementation

List view of WH related words from the documents shown below:

```
['keyword', 'methods', 'evaluated', 'criteria', 'usually', 'improves', 'selected', 'keywords', 'closer', 'ones',
'suggested', 'person', 'methods', 'ones', 'closest', 'human', 'point', 'evaluation', 'function', 'inspired', 'total',
'probability', 'employed', 'improve', 'farsi', 'automatic', 'attention', 'attractive', 'strings', 'proposed', 'evaluates',
'candidate', 'keywords', 'acquired', 'methods', 'ones', 'closest', 'human', 'point', 'attractive', 'strings', 'selected']
```

Figure 6.15 WH relations

CHAPTER 7

7.1 Implementation of keywords extraction

We select 8 keywords from the documents. By making the relation between Nouns to Nouns and Words, the whole implementation stage is ready to extract keywords from documents which denotes the depth of a text document.

Procedure of selecting keywords are given below:

- 3 keywords select from the Nouns to Nouns relation:

```
NounExtractorSize = sorted(NounExtractorSize.iteritems(),key=lambda (k,v): (v,k),reverse=True)

KeywordFinal = list()
Takenlist = list()
cnt = 0
for W in NounExtractorSize:
    Takenlist.append(W[0])
    if len(KeywordFinal) > 0:
        #print W[0]
        KeywordFinal.append(W[0])
        cnt = cnt + 1
        for check in KeywordFinal:
            if check in N_grams:
                to = N_grams[check]
                if to in KeywordFinal:
                    KeywordFinal.remove(to)
                    KeywordFinal.remove(check)
                    cnt = cnt - 2
                    KeywordFinal.append(check+" "+to)
                    cnt = cnt + 1
                    #print "Inside loop"



    else:
        #print W[0]
        KeywordFinal.append(W[0])
        cnt = cnt + 1
    #print cnt
    if cnt == 3:
        break
```

Figure 7.1 Implementation of 3 Keywords selection from Nouns to Nouns relation

❖ 3 keywords select from the Nouns to Words relation:

```
ResWordTF = dict()
for tak in MainAlgo:
    for tak1 in MainAlgo[tak]:
        ResWordTF[tak1] = ResWordTF.get(tak1, 0) + 1
ResWordTF = sorted(ResWordTF.iteritems(), key=lambda (k, v): (v, k), reverse=True)
cnt = 0
for W in ResWordTF:
    take = Snow_Stemmer.stem(W[0])
    if take in Takenlist:
        continue
    else:
        if W[0] in KeywordFinal:
            continue
        else:
            if W[0] == 'PH':
                continue
            else:
                Takenlist.append(W[0])
                if len(KeywordFinal) > 0:
                    # print W[0]
                    KeywordFinal.append(W[0])
                    cnt = cnt + 1
                    for check in KeywordFinal:
                        if check in N_grams:
                            to = N_grams[check]
                            if to in KeywordFinal:
                                KeywordFinal.remove(to)
                                KeywordFinal.remove(check)
                                cnt = cnt - 2
                                KeywordFinal.append(check + " " + to)
                                cnt = cnt + 1
                                # print "Inside loop"
                else:
                    KeywordFinal.append(W[0])
                    cnt = cnt + 1
    if cnt == 3:
        break
```

Figure 7.2 Implementation of 3 Keywords selection from Nouns to Words relation

❖ 2 keywords selection from WH questions relations:

```
ct = 0
print questionsTF
track = 0
for i in questionsTF:
    take = Snow_Stemmer.stem(i[0])
    if take in Takenlist:
        continue
    else:
        if take in KeywordFinal:
            continue
        else:
            if i[0] in All_Tagged:

                if All_Tagged[i[0]] == 'JJ':
                    track = track +1
                    KeywordFinal.append(i[0])
                    ct = ct + 1
                    if ct == 2:
                        break
```

Figure 7.3 Implementation of 2 Keywords selection from WH questions relation

If there is WH question in the documents then rest 2 words will be select from Nouns to Words relation and we prioritized the POS-tagging as NN>JJ>VBN for this thesis implementation.

7.2 Results

In this section, we will show some documents and their corresponding keyword extraction results using our selected approach.

The performance of automatic keyword extraction methods is evaluated by recall and precision criteria.
The result of keyword extraction usually improves when the selected keywords get closer to the ones suggested by a person. Since recall and precision has mutual effect on each other, increase in precision leads to recall decrease and vice versa. In this paper, a post-processing phase based on using attention attractive strings is proposed, which evaluates candidate keywords acquired from other keyword extraction methods to find the ones closest to the human point of view. In other words, an innovative keyword evaluation function, which is inspired by total probability theorem, is employed to improve the precision criterion in a Farsi automatic keyword extraction task. The attention attractive strings are selected by a reverse-engineering process from 800 Farsi keywordassigned documents. Results indicate that apart from extraction method, improving precision criterion is possible using the proposed post-processing phase without noticeable decrement in recall criterion.

Figure 7.4 Document-1

Breadth-first search is a simple strategy in which the root node is expanded first, then all the
SEARCH
successors of the root node are expanded next, then their successors, and so on. In general,
all the nodes are expanded at a given depth in the search tree before any nodes at the next
level are expanded.
Breadth-first search is an instance of the general graph-search algorithm in
which the shallowest unexpanded node is chosen for expansion. This is achieved very simply
by using a FIFO queue for the frontier. Thus, new nodes (which are always deeper than their
parents) go to the back of the queue, and old nodes, which are shallower than the new nodes,
get expanded first. There is one slight tweak on the general graph-search algorithm, which is
that the goal test is applied to each node when it is generated rather than when it is selected for
expansion. This decision is explained below, where we discuss time complexity. Note also
that the algorithm, following the general template for graph search, discards any new path to
a state already in the frontier or explored set; it is easy to see that any such path must be at
least as deep as the one already found. Thus, breadth-first search always has the shallowest
path to every node on the frontier.

Figure 7.5 Document-2

Harry Potter is a series of fantasy novels written by British author J K Rowling.
The novels chronicle the life of a young wizard, Harry Potter, and his friends Hermione Granger and Ron Weasley,
all of whom are students at Hogwarts School of Witchcraft and Wizardry.
The main story arc concerns Harry's struggle against Lord Voldemort,
a dark wizard who intends to become immortal, overthrow the wizard governing body known as the Ministry of Magic,
and subjugate all wizards and Muggles.

Figure 7.6 Document-3

lexical analysis splits input tokens purpose syntax analysis (also known parsing) recombine tokens
not list characters reflects structure text
something typically data structure called syntax tree text
indicates tree structure
leaves tree tokens lexical analysis leaves read left right sequence input text
hence important syntax tree leaves combined form structure tree interior nodes tree labelled

Figure 7.7 Document-4

Keyword extraction is highly related to automated text summarization. In text summarization, most indicative
sentences are extracted to represent the text. In keyword extraction, most indicative keywords are
extracted to represent the text. In both of these problems, the features like word frequencies, cue phrases, position
in text, lexical chains and discourse structure are exploited to discover a pattern representing importance
in a text. In this paper, we aim to explore the effect of lexical chains in keyword extraction, when the problem
is treated as a supervised machine learning task. This learning task uses features based on the lexical chains of
words. Since we can build lexical chains for words only (not for phrases) using the WordNet ontology (Fellbaum,
1998), we concentrate on the keyword extraction problem instead of keyphrase extraction.
Although we have experimented with different classifiers such as Naive Bayes, we obtained better results
with the decision tree induction algorithm C4.5 (Quinlan, 1993). For this reason, we have used C4.5 in order
to represent the keyword extraction problem as a learning task. We used C4.5 with two different sets of features.
In our baseline system, we used only the text features (without using any feature based on the lexical
chains of words). In the second case, C4.5 was used with the features based on the lexical chains in addition
to the features used in the baseline system. Then we compare the results of these two versions. We have
obtained better results when the features based on the lexical chains were used.

Figure 7.8 Document-5

Table 7.1 Important keywords extraction from documents

Import Keywords from Documents				
Document-1	Document-2	Document-3	Document-4	Dcoument-5
precision	search	wizard	structure	reason
keyword extraction	algorithm	harry	analysis leaves	keyword
phase	tweak	potter	syntax	lexical chains
methods ones	expanded	muggles	tokens lexical	c used
farsi	set	novels	input text	features based
proposed	selected	students	sequence	words
human	new	hermione	lexical	lexical
attractive	general	british	right	supervised

In the selection of keywords, we used N-grams(n==2) which gives a phrase of two words and we consider it as a keyword in process of selecting 8 keywords from the documents.

7.3 Comparison with other approaches or Toolkits

We named our approach as JEx and in this section we are going to compare our results with other toolkits. In the comparison, every approach use POS-tagging, stemming, TF terms. Cortical.io used semantic fingerprint to denote the identity card of a single concept. It characterizes in a unique, descriptive way the meanings associated with that concept. In that approach, words are being converted into their semantic fingerprints and it is unique for any words, documents etc. In our thesis work, we developed a new approach and it is called as semantic relation between Nouns to Nouns and Words along with other NLP tasks.

Table 7.2 JEx vs NLTK vs Cortical.io vs keywordextraction.net toolkit on Document-1:

Import Keywords from Document-1			
JEx	NLTK	Cortical.io	keywordextraction.net
precision	extraction	keyword	keyword extraction
keyword extraction	precision	precision	extraction method
phase	keywords	extraction	evaluates candidate
methods ones	methods	recall	probability theorem
farsi	attractive	methods	automatic keyword
proposed	automatic	Phase	extraction task
human	improve	ones	evaluation function
attractive	usually	criterion	automatic keyword

Table 7.3 JEx vs NLTK vs Cortical.io vs keywordextraction.net toolkit on Document-2:

Import Keywords from Document-2			
JEx	NLTK	Cortical.io	keywordextraction.net
search	search	node	goal test
algorithm	node	nodes	slight tweak
tweak	path	search	new path
expanded	general	algorithm	nodes
set	breadth-first	queue	root nodes
selected	new	path	such path
new	expanded	root	general template
general	nodes	complexity	graph search algorithm

Table 7.4 JEx vs NLTK vs Cortical.io vs keywordextraction.net toolkit on Document-3:

Import Keywords from Document-3			
JEx	NLTK	Cortical.io	keywordextraction.net
wizard	wizard	harry potter	main story
harry	potter	rowling	J k rowling
potter	ron	novels	hermione granger
muggles	novels	wizards	young wizard
novels	young	hermione	novels chronicle
students	overthrow	magic	struggle
hermione	main	wizardry	british author
british	novels	witchcraft	fantasy novels

Table 7.5 JEx vs NLTK vs Cortical.io vs keywordextraction.net toolkit on Document-4:

Import Keywords from Document-4			
JEx	NLTK	Cortical.io	keywordextraction.net
structure	structure	syntax	lexical analysis
analysis leaves	text	input	tree structure
syntax	syntax	tree	list characters
tokens lexical	tokens	parsing	important syntax tree
input text	lexical	tokens	syntax tree text
sequence	right	structure	form structure tree
lexical	interior	analysis	interior nodes tree
right	leaves	nodes	right sequence

Table 7.6 JEx vs NLTK vs Cortical.io vs keywordextraction.net toolkit on Document-5:

Import Keywords from Document-5			
JEx	NLTK	Cortical.io	keywordextraction.net
reason	text	keyword	decision tree
keyword	extraction	extraction	learning task uses
lexical chain	keyword	words	machine learning
c used	features	phrases	keyword extraction
feature based	lexical	chains	learning task
words	different	baseline	extraction problem
lexical	indicative	problem	text summarization
supervised	build	results	keyphrase extraction

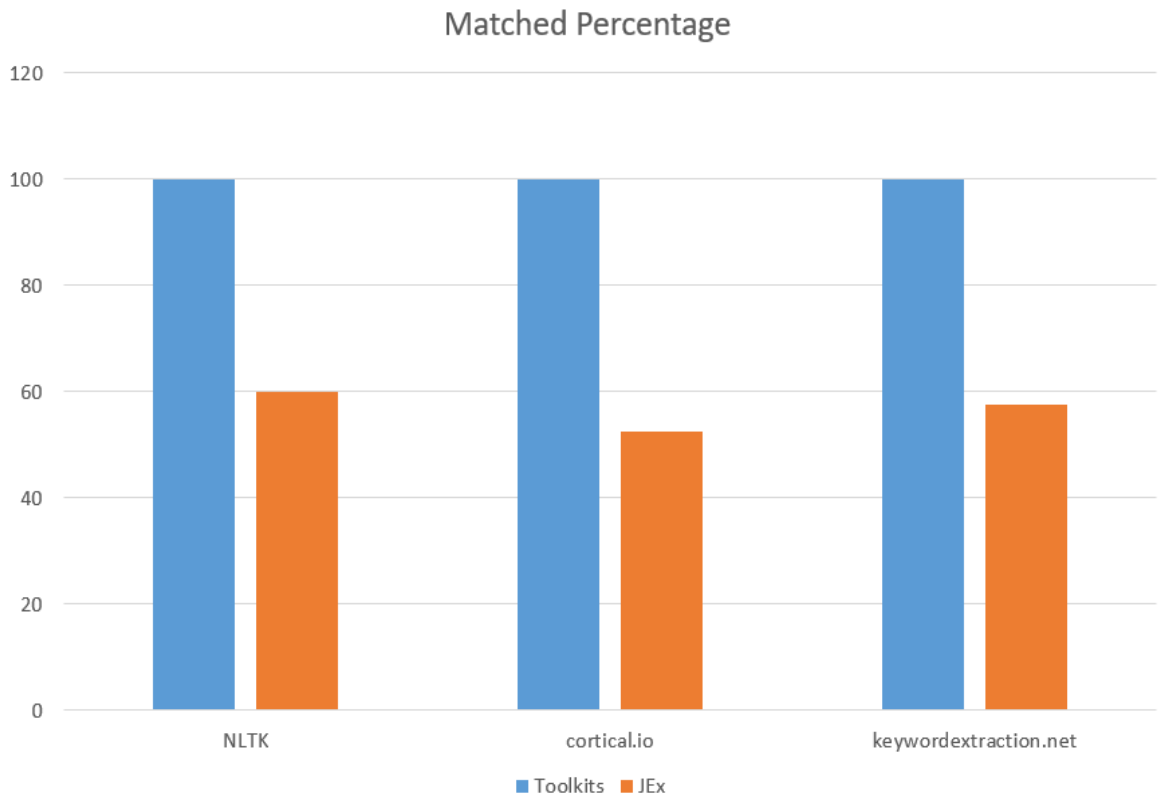

Figure 7.10 Percent of Matched Keywords

Analyzing the comparison between 3 selected toolkits with our approach, we have found that matching keywords ratio is around 60%. This is a wide fields where keywords could be different depending on the text. Some keywords might show the exact depth of the documents as well as some other words might do the same. So we could say that our approaches shows convincing results by the statistics of comparison with 3 different approaches along with ours.

CHAPTER 8

8.1 Conclusion and Future Work

The aim of this research is to find out the important keywords from the documents to help us save a huge amount of time from doing unusual research behind massive documents. We became successful to achieve our aim and a completely different approach has been presented in the report. The key features of our research are the specific use of Term frequency, POS-tagging, Semantic relation between words and sentences, and other NLP basic modules. The highlights of the major outcomes are that Semantic relation is a productive approach, can be utilized to discover the important keywords from the documents.

This research work can be further extended to identify the keywords which could be more accurate to detect the content of a document. In our research, we used term frequency but TF-IDF could be excellent use of modules to extract from the documents. We selected the Nouns as a heart of the sentence but Verb and Adjective also plays a big role. In future, we will consider TF-IDF, verb and Adjective along with the present approach to improve accuracy of finding important keywords from the documents.

REFERENCES

[1] C. Zhang, H. Wang, Y. Liu, D. Wu, Y. Liao, B. Wang, Automatic "Keyword extraction from documents using conditional random fields", J. Comput. Inf. Syst. 3 (2008)

[2] H. H. Kian · M. Zahedi, "Improving Precision in Automatic Keyword Extraction Using Attention Attractive Strings", Arab J SciEng DOI 10.1007/s13369-013-0573-6

[3] A. Hulth, "Improved Automatic Keyword Extraction Given More Linguistic Knowledge"

[4] H. Zhao, C., Song Zhu, "Automatic Keyword Extraction Algorithm and Implementation", Applied Mechanics and Materials Vols 44-47 (2011) pp 4041-4049 Online: 2010-12-06 © (2011) Trans Tech Publications, Switzerland doi:10.4028/www.scientific.net/AMM.44-47.4041

[5] G. Ercan, I. Cicekli, "Using lexical chains for keyword extraction", Information Processing and Management 43 (2007) 1705–1714

[6] Y. Matsuo, M. Ishizuka, "Keyword Extraction From A Single Document Using Word Co-occurrence Statistical Information", International Journal on Artificial Intelligence Tools Vol. 13, No. 1 (2004) 157169

[7] R. Kongkachandra; K. Chamnongthai, "Abductive Reasoning for Keyword Recovering in Semantic-based Keyword Extraction", Fifth International Conference on Information Technology: New Generations

[9] Y.B. Wu; Q. Li; R. S. Bot; X. Chen, "Domain-specific Keyphrase Extraction", Information Systems Department New Jersey Institute of Technology Newark, NJ 07102.

[10] A. Dutta, "A Novel Extension for Automatic Keyword Extraction", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 6, Issue 5, May 2016

[11] H. Haggag; A. Abutabl; A. Basil, "Keyword Extraction using Clustering and Semantic Analysis", International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064 Impact Factor (2012): 3.358

[12] H. Haggag, "Keyword Extraction using Semantic Analysis", International Journal of Computer Applications (0975 – 8887) Volume 61– No.1, January 2013

[13] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge", In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'03), 216 – 223, Sapporo, 2003

[14] Y. Suzuki, F. Fukumoto, Y. Sekiguchi, "Keyword extraction of radio news using term weighting with an encyclopedia and newspaper articles", Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, 373 - 374, New York, USA, 1998.

[15] M. Andrade and A. Valencia, “Automatic extraction of keywords from scientific text: application to the knowledge domain of protein families”, Bioinformatics, 1998, 14(7), 600–607.

[16] A. Hulth , “Combining machine learning and natural language processing for automatic keyword extraction”, PhD Thesis, Stockholm University, Faculty of Social Sciences, Department of Computer and Systems Sciences, 2004

[17] L. F. Chien, “PAT-Tree-Based Keyword Extraction for Chinese Information Retrieval”, Proceedings of the ACM SIGIR International Conference on Information Retrieval, 1997, pp. 50--59.

[18] B. Hong, D. Zhen, “An Extended Keyword Extraction Method”, 2012 International Conference on Applied Physics and Industrial Engineering

[19] A. Azcarraga, M. Liu, R. Setiono, “Keyword Extraction Using Backpropagation Neural Networks and Rule Extraction”, WCCI 2012 IEEE World Congress on Computational Intelligence June, 10-15, 2012 - Brisbane, Australia

[20] "Cortical.io - Fast, precise, intuitive NLP", Cortical.io, 2017. [Online]. Available: http://www.cortical.io/. [Accessed: 07- Apr- 2017]

[21] "Keyword Extractor Online Demo - Keyword Extraction | Keyphrase Extraction | Keyword Extraction Online | Keyword Extraction Demo | Keyword Extraction API", Keywordextraction.net, 2017. [Online]. Available: http://keywordextraction.net/keyword-extractor-online-demo. [Accessed: 07- Apr- 2017]

[22] "Natural Language Toolkit — NLTK 3.0 documentation", Nltk.org, 2017. [Online]. Available: http://www.nltk.org/. [Accessed: 07- Apr- 2017]

[23] "TextBlob: Simplified Text Processing — TextBlob 0.12.0 documentation", Textblob.readthedocs.io, 2017. [Online]. Available: https://textblob.readthedocs.io/en/dev/. [Accessed: 07- Apr- 2017]

[24] Turney, P.D.: Learning to extract keyphrases from text. NRC Technical ReportERB-1057, pp. 1–43. NationalResearch Council,Canada (1999)

[25] Zhang, K., Xu, H.; Tang, J.; Li, J.: Keyword extraction using support vector machine. In: Advances in Web-Age Information Management, pp. 85–96 (2006)

[26] E. Frank, G. W. Paynter, I. H. Witten. Domain-Specific Keyphrase Extraction. In: Proceedings of the 16th International Joint Conference on Aritifcal Intelliegence, Stockholm, Sweden, Morgan Kaufmann, 1999: 668-673.

[27] Teufel, S., & Moens, M. (1997). Sentence extraction as a classification task. In Inderjeet Mani & Mark T. Maybury (Eds.), Proceedings of ACL/EACL97-WS. Spain: Madrid.