Automatic *Keyword* Extraction for Text Summarization in e-Newspapers

Justine Raju Thomas
Department of Computer Science and Engineering
National Institute of Technology
Rourkela - 769008, Odisha, India
justine.thomas@snapdeal.com

Santosh Kumar Bharti
Department of Computer Science and Engineering
National Institute of Technology
Rourkela - 769008, Odisha, India
sbharti1984@gmail.com

Korra Sathya Babu
Department of Computer Science and Engineering
National Institute of Technology
Rourkela - 769008, Odisha, India
prof.ksb@gmail.com

ABSTRACT

Summarization is the process of reducing a text document to create a summary that retains the most important points of the original document. Extractive summarizers work on the given text to extract sentences that best convey the message hidden in the text. Most extractive summarization techniques revolve around the concept of finding keywords and extracting sentences that have more keywords than the rest. Keyword extraction usually is done by extracting relevant words having a higher frequency than others, with stress on important ones'. Manual extraction or annotation of keywords is a tedious process brimming with errors involving lots of manual effort and time. In this paper, we proposed an algorithm to extract keyword automatically for text summarization in e-newspaper datasets. The proposed algorithm is compared with the experimental result of articles having the similar title in four different e-Newspapers to check the similarity and consistency in summarized results.

CCS Concepts

•**Information systems** → **Data mining;** *Data analytics;*

Keywords

Automatic keyword detection, e-Newspaper, Natural language processing, Text summarization.

1. INTRODUCTION

There are many popular e-newspapers available freely in the internet, such as, Times of India, The Hindu, Hindustan Times, Indian Express, etc. People find a lot of information every day in news articles. Extracting all the relevant information from these newspapers is a tedious job for individuals. There is a need for a tool that extracts only needed information from these data sources. To get the required information, we need to mine the text from newspapers. Text mining is the process of extracting large quantities of text to derive high-quality information. Natural language processing (NLP) is a powerful tool for text mining. Text mining deploys some of the techniques of NLP (such as parts-of-speech (POS) tagging, parsing, N-grams, tokenization, etc.) to perform analysis. Text mining includes tasks like automatic keyword extraction and text summarization.

Automatic keyword extraction is the process of selecting words and phrases from an article that can at best project the core sentiment of the article without any human intervention depending on the model [26]. The target of automatic keyword extraction is the application of the power and speed of current computation abilities to the problem of access and recovery, stressing upon information organization without the added costs of human annotators.

Summarization is a process where the most salient features of a text are extracted and compiled into a short abstract of the original wording [11]. According to Mani and Maybury [17], text summarization is the process of "distilling the most important information from a text to produce an abridged version for a particular task and user". Summaries are usually around 17% of the original text and yet contain everything that could have been learnt from reading the original article [8]. In the wake of big data analysis, summarization is an efficient and powerful technique to give a glimpse of the data. It can be observed that summarization rely heavily on keyword extraction. Therefore, we focused our attention on the integration between them.

In this paper, an algorithm is proposed for automatic keyword extraction for text summarization. It is capable of handling the limitation of existing techniques like adding a stop list to extract important keyword which might include words that are essential to the article and might lead some relevant words to lose its significance. In addition the rarity of the word in other texts was also considered. The word which is highly frequent in the article under review and is hardly found in other documents gets very high scores so that only the words pertinent to this article gets chosen as keywords. The proposed algorithm follows a mixed approaches of machine learning and statistical method. A hidden markov model (HMM)- based POS tagger [1] is used to identify POS information of an article and then a statistical method is used to extract keywords. The algorithm for automatic keyword extraction use a learned probability

ICIA-16, August 25-26, 2016, Pondicherry, India
© 2016 ACM. ISBN 978-1-4503-4756-3/16/08...$15.00
DOI: http://dx.doi.org/10.1145/2980258.2980442

distribution to assign scores to each word. The keywords for the article under consideration are determined based on these scores and are used to summarize the article. The summarization algorithm accordingly selects sentences to form the required summary. This algorithm applies to a single article at a time.

The rest of the paper is organized as follows: Related work is presented in Section 2. Preliminary of the paper is given in Section 3. Section 4 describes the proposed scheme and implementation details. Experimental results are shown in Section 5. Finally, conclusion of the paper given in Section 6.

2. RELATED WORK

In recent times, many authors suggested the procedure for automatic keyword extraction in their state-of-the-art work [24, 10, 7, 15]. On the basis of previous work done towards automatic keyword extraction from the text for its summarization, extraction techniques can be classified into four categories, namely, simple statistical approach, linguistics approach, machine learning approach, and hybrid approaches [26] as soon in Figure 1.

2.1 Simple Statistical Approach

These methods are crude, simplistic and tend to have no training sets. They focus on statistics derived from non-linguistic features of the document text such as the position of a word within the document, the term frequency, and inverse document frequency. These statistics are later used to develop a list of keywords. Cohen [5], used n-gram statistical information to find the keywords within a document automatically. Other methods within this category include word frequency, term frequency (TF) [16] or term frequency-inverse document frequency (TF-IDF) [21], word co-occurrences [19], and PAT-tree [4]. The most fundamental of them is term frequency. In these methods, the frequency of occurrence is the only criteria that decide whether a word is a keyword or not. It is very crude and tends to give quite inappropriate results. An improvement of this method is the TF-IDF, which also takes the frequency of occurrence of a word as the criterion to decide a keyword or not. Similarly, word co-occurrence algorithms deal with statistical information about the number of times a word has occurred and the number of times it has happened with another word. This statistical information is then used to calculate support and confidence of the words. Apriori method is then used to derive the keywords.

2.2 Linguistics Approach

This approach use the linguistic features of the words in the sentences and documents. The linguistics approach includes the lexical analysis [2], syntactic analysis [12], discourse analysis [22], etc. Electronic dictionary, Tree Tagger, WordNet, n-grams, POS pattern, etc., are the resources of lexical analysis while noun phrase (NP) chunks (Parsing) belong to syntactic analysis.

2.3 Machine Learning Approach

Keyword extraction can also be seen as a learning problem. This approach requires manually annotated training data and training models. Hidden markov model [6], support vector machine (SVM) [27], naive Bayes (NB) [9], bagging [12], etc. are commonly used training models in these approaches. In the second phase, the document whose keywords are to be extracted are gives as inputs to the model, which then extracts the keywords that best fit the model's training. One of the most famous algorithms in this approach is the KEA [25]. The article is first converted into a graph. Each word is treated as a node, and whenever two words appear in the same sentence, the nodes are connected with an edge for each time they appear together. Then the number of edges connecting the vertices are converted into scores and are clustered accordingly. The cluster heads are treated as keywords. Bayesian algorithms use the Bayes classifier to classify the word into two categories: keyword or not a keyword depending on how it is trained. GenEx [23] are other tools in this approach.

2.4 Hybrid Approach

These approaches combine the above two methods or use heuristics, such as position, length, layout feature of the words, HTML tags around the words, etc. [13]. These algorithms are designed to take the best features from above mentioned approaches.

Based on the classification shown in Figure 1, we bring a consolidated summary of previous studies on automatic keyword extraction is given in Table 1. It discuss the approaches that are used for keyword extraction, various datasets in different domains and experiment performed using single or multiple documents. Table 2 shows the details about approaches, types of document and different domain for keyword extraction.

3. PRELIMINARIES

The focus of our work lies in enabling a user to search for keywords from within a text file and get the summary of the entire document using those keywords. To achieved this, POS tagging, anaphora and cataphora resolution preliminaries are used and are explained below

3.1 POS Tagging

It is a process of taking a word from the text (corpus) as input and assign corresponding part-of-speech to each word as output based on its definition and context *i.e.* relationship with adjacent and related words in a phrase, sentence, or paragraph. In this paper, an HMM-based POS tagger is deployed to identify correct POS information of words in given sentences or phrases. For example POS tag for the sentence, " Love has no finite coverage" is love|NN, has|VBZ, no|DT, finite|JJ, coverage|NN. In this work, Penn Treebank tag set notations [18] are followed. It is a brown corpus style of tagging having 44 tags. Such as JJ-adjective, NN-noun, RB-adverb, VB-verb, and UH-interjection, etc.

3.2 Hidden Markov Model

Hidden markov model is a supervised machine learning classifier. The automatic labeling of words to their parts of speech is known as POS tagging. POS tagging is a supervised learning problem. Therefore, HMM [1] is used to analyze the POS tags for a given text. With the help of HMM, we can resolve the ambiguity problem in POS tagging such as "I **love** writing" and "**Love** has no finite coverage." In the sentence "I **love** writing", "love" act as a verb (VB) while in "Love has no finite coverage", "love" act as a noun (NN). HMM analyzes this ambiguity correctly and gives accurate POS information as an output.

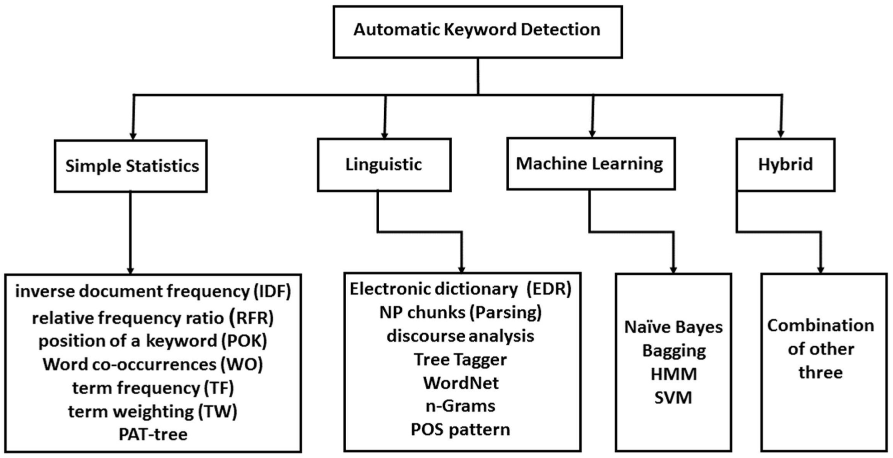

Figure 1: Classification of automatic keyword extraction on the basis of approaches used in existing literature

3.3 Anaphora and Cataphora Resolution

The word anaphora is derived from an ancient Greek word which means "the act of carrying back upstream". It involves two parts; anaphor: the part which is pointing (reference) and the antecedent: the part which is being pointed to. For example: "Sachin isn't out yet, but he should be any minute". In this example, Sachin is the antecedent, and he is the anaphor. Anaphora resolution deals with matching the anaphor to the antecedent. Hence converting the above sentence into Sachin isn't out yet but Sachin should be any minute.

Similarly cataphora too is derived from the ancient Greek word, meaning "the act of carrying forward". It involves replacing an anaphor by an antecedent that occurs later in the text and not before as was the case in anaphora resolution. For example: "It is tough, It is irritating, and it is annoying. I hate writing the thesis". Cataphora resolution would resolve the above sentence into "writing the thesis is tough, writing the thesis is irritating and writing the thesis is annoying. I hate writing the thesis". This paper followed [14] algorithm for anaphora and cataphora resolution in english text.

4. PROPOSED SCHEME

This section deals with explicit details on the approach that was used for automatic keyword extraction and summarization.

4.1 Automatic Keyword Extraction

The primary aim of automatic keyword extraction is to point out a set of words or phrases that best represents the document. To achieved this, a hybrid extraction technique has been proposed. The steps are shown in Figure 2.

4.1.1 Keyword Annotation

For this model, there is a need of human intervention for an annotation to train the proposed algorithm. The human annotators analyze documents and select probable keywords. These keywords are supplied to the POS tagger on the documents and the output is provided to the next section of the model.

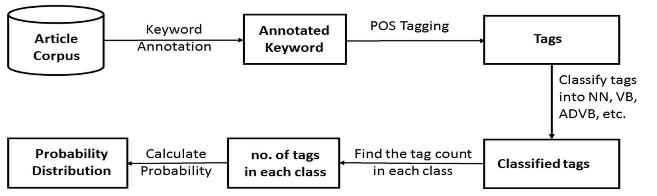

Figure 2: Training model for automatic keyword extraction

4.1.2 HMM-based POS Tagging

In this paper, we have used an HMM-based POS tagger [1] to analyze accurate POS information of any given text based on its context (relationship with adjacent and related words in a phrase, sentence, or paragraph).

4.1.3 Learning Probability Distribution

Due to lack of reliable human annotators, e-newspapers clippings are used as our training dataset. The articles were considered as the target document and the headlines as the keywords, thus eliminating the need of human annotators. The training dataset was analyzed and found the number of nouns, verbs, adverbs, adjectives, etc. appeared as a keyword in the headlines. The algorithm for finding probability distribution values is given in Algorithm 1.

Algorithm 1 derives P(tag) from the dataset. Input provided by the algorithm are the dataset of articles along with human annotated keywords for each article. Initially, the variable count value is initialized to 0. This variable stores the number of keywords that has been scanned by the algorithm. First while loop initializes the tag count for every POS tag to 0. Next, two while loops find POS tag of the keyword for every keyword in every article of the dataset and the count for that POS tag is increased by 1. Once this terminates, probability distribution, (P(tag)) is determined by dividing the tag count by the total number of keywords. This is used as a probabilistic measure to detect keywords.

4.2 Extraction

Extraction (testing) model is shown in Figure 3. It re-

Table 1: Previous studies in automatic keyword extraction for text summarization.

Study	Types of Approach				Doc. Type		Domain Type						
	T1	T2	T3	T4	P1	P2	D1	D2	D3	D4	D5	D6	D7
Dennis *et al.*, 1967		✓			✓				✓				
Salton *et al.*, 1991		✓			✓							✓	
Cohen *et al.*, 1995	✓				✓			✓					
Chien *et al.*, 1997	✓				✓			✓					
Salton *et al.*, 1997		✓			✓							✓	
Ohsawa *et al.*, 1998	✓				✓			✓					
Hovy *et al.*, 1998		✓			✓				✓				
Fukumoto *et al.*, 1998	✓				✓		✓		✓			✓	
Mani *et al.*, 1999		✓			✓								
Witten *et al.*, 1999			✓		✓					✓			
Frank *et al.*, 1999			✓		✓			✓		✓			
Barzilay *et al.*, 1999		✓			✓								
Turney *et al.*, 1999			✓		✓			✓					
Conroy *et al.*, 2001			✓		✓					✓			
Humphreys *et al.*, 2002		✓		✓	✓	✓							✓
Hulth *et al.*, 2003		✓	✓		✓			✓		✓			✓
Ramos *et al.* 2003	✓				✓								
Matsuo *et al.*, 2004	✓				✓								
Erkan *et al.*, 2004		✓			✓								
Van *et al.*, 2004		✓			✓						✓		
Mihalcea *et al.*, 2004			✓		✓				✓				
Zhang *et al.*, 2006			✓		✓			✓					
Ercan *et al.*, 2007			✓		✓			✓					
Litvak *et al.*, 2008			✓		✓								✓
Zhang *et al.*, 2008			✓		✓			✓					

Table 2: Types of approaches, document and domain used

	Types of Approaches
T1	Simple Statistics (SS)
T2	Linguistics (L)
T3	Machine Learning (ML)
T4	Hybrid (H)
	Types of Document
P1	Single Document (SD)
P2	Multiple Document (MD)
	Types of Domain
D1	Radio News (RN)
D2	Journal Articles (JA)
D3	Newspaper Articles (NA)
D4	Technical Reports (TR)
D5	Transcription Dialogues (TD)
D6	Encyclopedia Article (EA)
D7	Web Pages (WP)

quires anaphora and cataphora resolution to pre-process the article. These pre-processed articles are supplied to the POS tagger on the documents. The score is calculated for each text and few top scored texts are selected as a keyword.

4.2.1 Keyword Extraction

The output file from the POS tagger is now forwarded to the model for extraction. Unlike tf-idf (keeping the count of the number of times a particular word has appeared) we keep count of the word-tag pair. *i.e* [Can, Noun] and [Can, Verb] are treated differently. When a count of the entire document is taken, the keywords are ranked by Equation 1.

$$Score = P(tag) * Count(word, tag) \qquad (1)$$

Algorithm 1: Algorithm to Train Probability Distribution

```
Data: dataset := dataset of articles
      keyword := Human annotated set of keywords
for each article.
Result: P(tag) : Probability Distribution of Tags
count=0
while tag in taglist do
    Tag_count(tag)=0
end
while article in dataset do
    while keyword in article do
        tag=POS_tag(keyword)
        Tag_count(tag)=Tag_count(tag)+1
        count=count+1
    end
end
while tag in taglist do
    P(tag)=Tag_count(tag)/count
end
```

where, *P(tag)* is the probability of a tag being a keyword and *count(word, tag)* is the number of times the word has appeared in the current document. P(tag) is the same trained probability measure that is explained in Algorithm 1.

Algorithm 2 takes single document article, the number of keywords to be extracted and a probability distribution table trained during the training as an input for extracting keywords. The output of the algorithm will be saved in an array Keywords[]. Wordset[] is another array of structures that keeps the record of the words that have already been scanned and number of times that word-tag pair has been scanned. The top is the variable that stores the value

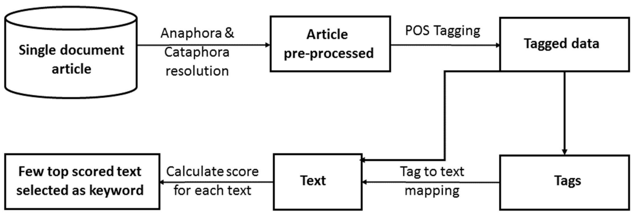
Single document article
Anaphora & Cataphora resolution
Article pre-processed
POS Tagging
Tagged data
Tags
Tag to text mapping
Text
Calculate score for each text
Few top scored text selected as keyword

Figure 3: Testing model for keywords extraction
l

of the number of words scanned and it is initialized to 0. The input file is passed through an anaphora and cataphora resolver. The output file of anaphora and cataphora resolver is subjected to POS tagger to get the POS tag values. The algorithm then courses through the file, updating existing records in the way and creating new ones when needed. When the algorithm is done with parsing the file, the scores are updated. Once the scores are set, the array is sorted according to the scores of each word-tag pair. The top score value of few texts are then extracted as keywords.

5. SUMMARIZATION

With the help of algorithms explained so far, a set of word tag pair keywords are attained as well as their respective scores. For summarization, the proposed algorithm suggests that one derives from many sentences for a keyword from the article as is proportional to the score it received. It can derive these sentences by any means, be it through clustering means or crude scoring. The added advantage of this algorithm is the simple statement that "Not all keywords are equal". So it helps while selecting the keywords by differentiating them.
Finally, let us see an example of the working procedure of the proposed scheme. Let us assume a single document article on pollution. Possible keywords would be pollution, destruction, harmful and disease. Suppose one need to summarize it in twenty sentences. Given the individual scores of the keywords as shown in Table 5, we shall extract sentences for each of the keywords using Equation 2. Finally, the extracted number of sentences is shown in Table 6 to get the desired summary of the document.

$$DSIA = \lceil \frac{(Keyword\ score * No.\,of\ sentences\ req.)}{(Total\ score\,of\ all\,the\,keywords)} \rceil \quad (2)$$

Where, DSIA - desired number of sentences in summary using each keyword.

6. RESULTS & DISCUSSION

This section evaluates the quality of the keywords produced via our suggested algorithms. The section starts with article collection from different e-Newspapers followed by results and discussion. Finally, we compare the result of proposed algorithm with the state-of-the-art.

6.1 Article Collection

After studying the website of several e-Newspapers, we collected data from four different e-Newspapers namely, Times of India, The Hindu, Hindustan Times, Indian Express. Our dataset included almost 600 articles from each e-Newspapers

Algorithm 2: Extract_Keywords

```
Data: $doc$ := Input Article
        P(Tag) := Set of Trained Probabilities
        $Num\_Keywords$ := Required Number of Keywords
Result: $Keywords[]$
$res\_doc$:=resolve($doc$)
$pos\_doc$:=pos_tagger($res\_doc$)
$top$:=0
while $word$ in $pos\_doc$ do
    $flag$:=0
    for $i \leftarrow 0$ to $top$ do
        if $word.text$=$wordset[i].text$ and $word.tag$=$wordset[i].tag$ then
            $wordset[i].count$:=$wordset[i].count$+1
            $flag$:=1
        end
    end
    if $flag$=0 then
        $wordset[top+1].word$:=$word.word$
        $wordset[top+1].tag$:=$word.tag$
        $wordset[top+1].count$:=1
        $wordset[top+1].score$:=0 $top$:=$top$+1
    end
end
for $i \leftarrow 0$ to $size$ do
    $wordset[i].score$:=$wordset[i].count$*P($wordset[i].tag$)
end
sort_desc($wordset.score$)
for $i \leftarrow 0$ to $Num\_Keywords$ do
    $Keywords[i]$:=$wordset[i]$
end
```

Table 3: Score of Each Keywords

Pollution	Destruction	Harmful	Disease
4.3	0.8	2.4	2.5

ranging from the 1st of January 2016 to 15th of March 2016. Number of articles collected from each e-Newspapers are as follows:

- A dataset of 600 articles - a total of 2678 keywords identified (Human Annotated) from Times of India epaper.
- A dataset of 589 articles - a total of 2598 keywords identified (Human Annotated) from The Hindu epaper.
- A dataset of 612 articles - a total of 2638 keywords identified (Human Annotated) from Hindustan Times epaper.
- A dataset of 570 articles - a total of 2568 keywords identified (Human Annotated) from Indian Express epaper.

Table 4: Required Number of Sentences on Each Keyword

Pollution	Destruction	Harmful	Disease
9	2	5	5

6.2 Experimental Results and comparison

Algorithm 3 finds the count value of each keyword as shown in Table 5 and convert them into probabilistic values, by dividing the counts by 2678, 2598, 2638, 2568 respectively, which gives us a trained probabilistic measure as shown in Table 6.

To evaluate the performance of proposed algorithm and compare the results with existing work, three parameters are considered, namely, *precision*, *recall* and $f - score$. The formula to calculate *precision*, *recall* and $f - score$ shown in Equations 3, 4, 5 respectively.

$$Accuracy = \frac{T_p + T_n}{TNK} \tag{3}$$

$$Precision = \frac{T_p}{T_p + F_p} \tag{4}$$

$$Recall = \frac{T_p}{T_p + F_n} \tag{5}$$

$$F - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{6}$$

where,
TNK = Total Number of keyword, T_p = True positive, F_p = False positive, F_n = False negative

Precision is a statistical parameter that shows, how many items are correctly identified out of total identified items by the algorithm with the reference of ground truth. Correctly identified items are called true positive (T_p) as algorithm and ground truth, both identified positive, and the items are incorrectly identified called false positive (F_p) as algorithm identified positive but ground truth was negative. Similarly, how many of items correctly rejected by the algorithm is called true negative (T_n) as both algorithm and ground truth identified negative and how many items are incorrectly rejected called false negative (F_n) as algorithm identified negative but ground truth was positive.

For testing purposes, we ran our algorithm on a set of newspaper articles from same four e-Newspapers namely, Times of India, The Hindu, Hindustan Times, Indian Express. However, this time, the headlines were not provided to the algorithm. We collected the content of the article with similar article title in all four e-Newspapers (same news in all four e-Newspapers on the same date) to check the accuracy of proposed algorithm. The input was the same newspaper clippings, and the end target was to extract words that were present in the headlines. On having run the algorithm against the clippings, we got the following results shown in Table 7. Based on Table 9 results, calculated *precision*, *recall* and $f - score$ are shown in Table 8. The result of Table 8 shows how effectively our proposed algorithm works on all four e-Newspapers, and all of them attains almost similar *accuracy*, *precision*, *recall* and $f - score$ values. Finally, Table 9 shows the comparison of results against state-of-the-art. Higher *accuracy*, *precision*, *recall* and $f - score$ values indicated in bold.

6.3 Results Discussion

Just knowing the number of sentences to extract, is not sufficient. One also needs to know how to extract those sentences. The proposed algorithm is meant to piggyback upon other algorithms. Here, how our algorithm piggybacks upon the two existing algorithms that were discussed in related work.

6.4 Piggybacking on FCFS

Let us continue working with the example described above. The algorithm starts extracting sentences from the start of the document. We already know the set of keywords and how many sentences each keyword can bring into the summary. Now suppose, we finished extracting five sentences for the keyword 'Harmful'. The quota for that particular word is over. Hence, the word is removed from the keyword list. The algorithm continues until all the keywords are removed the keyword list.

6.5 Piggybacking on Text Rank - I

As explained earlier, the algorithm sorts the sentences in descending order according to the number of keywords each sentence contains. The algorithm now starts popping sentences from the top of this sorted stack. In our algorithm, instead of popping a predetermined number of sentences from the top, if a keyword is present in the popped sentence, the number of allocated sentence is reduced by 1. However if a sentence has no keywords whose allocated sentence is greater than 0, the sentence is rejected. The algorithm continues until the number of allocated sentence for each keyword becomes 0.

6.6 Piggybacking on Text Rank - II

This algorithm contains a slight twist to it. Instead of scoring sentences by the number of keywords it contains, we score them using the scores of the individual keywords present in them. For example: "Pollution is so harmful that many international agencies have come together to battle pollution". The above sentence has a score of 2*score (Pollution) + score (Harmful). The algorithm then sorts the sentences according to these scores in descending order and extracts the required number of sentences from the top.

Similarly, our algorithm can be piggybacked over numerous existing algorithms, which gives it a flexibility of use. The reason our algorithm works so good is that it rides on the firm foothold of tf-idf which is one of the highest-scoring performers. Adding to that, the dominant features of natural language processing made it perform better. The concept of anaphora and cataphora resolution too empowered the algorithm to perform better than the existing algorithms. Thus, we managed to take the most powerful features out of statistical and pattern recognition approaches to creating a robust hybrid algorithm. The demerits of this algorithm are that it is much more time taking than some of the existing algorithms.

The proposal on the text summarization algorithm introduces a new concept that encourages people to treat each word with a difference. Some keywords are more relevant to the topic, and some are less relevant to it. That fact has been exploited in the proposed algorithm and what makes it unusual is its flexibility to be piggybacked over most existing summarization algorithms.

7. CONCLUSION

This work has proposed interdependent algorithms in keyword detection and text summarization. The keyword detection algorithm worked very efficiently in recognizing key-

Table 5: Number of times a tag has been found in different e-Newspapers headlines

Name of e-Newpaper	VB	NN	VBD	NNP	NNS	JJ	VBG	JJS
Times of India	222	926	206	342	385	328	144	76
The Hindu	217	896	226	353	314	349	139	83
Hindustan Times	233	957	173	282	373	369	161	87
Indian Express	207	1002	183	284	364	298	149	86

Table 6: Probability measure for each tag has been found in different e-Newspapers headlines

Name of e-Newpaper	VB	NN	VBD	NNP	NNS	JJ	VBG	JJS
Times of India	0.082	0.363	0.077	0.128	0.144	0.122	0.053	0.028
The Hindu	0.083	0.344	0.087	0.136	0.121	0.134	0.053	0.032
Hindustan Times	0.088	0.362	0.065	0.106	0.141	0.140	0.061	0.033
Indian Express	0.080	0.390	0.071	0.110	0.141	0.116	0.058	0.033

Table 7: T_p, F_p, T_n and F_n of different e-Newspapers article headlines

e-Newspaper	$TotalKeyword$	T_p	F_p	T_n	F_n
Times of India	2678	178	119	2252	129
The Hindu	2598	158	101	2222	117
Hindustan Times	2638	168	113	2242	115
Indian Express	2568	163	104	2192	109

Table 8: $Accuracy$, $Precision$, $Recall$ **and** $F-score$ **of different e-Newspapers**

e-Newspapers	$Accuracy$	$Precision$	$Recall$	$F-score$
Times of India	0.907	0.599	0.579	0.588
The Hindu	0.916	0.610	0.574	0.591
Hindustan Times	0.913	0.597	0.593	0.594
Indian Express	0.917	0.610	0.599	0.604

Table 9: Comparison of proposed system result's with state-of-the-art

Approaches	$Precision$	$Recall$	$F-score$
Tf	0.53	0.48	0.50
Tf-Idf	0.55	0.61	0.58
KeyGraph	0.42	0.44	0.43
Lexical Chains	0.45	0.37	0.41
Word-Cooccurrence	0.51	0.62	0.56
Bayes	0.38	0.81	0.52
Proposed	**0.61**	0.57	**0.59**

words and had an impressive precision, recall and f-score over existing algorithm. Our proposed algorithm attains a precision of 0.61, a recall of 0.57 and an f-score of 0.59. By the keyword extraction algorithm, a summarization algorithm was proposed that introduced a concept that said "not all words are equal", and yet had the flexibility to piggyback over other existing summarization algorithms.

References

[1] M. Banko and R. C. Moore. Part of speech tagging in context. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[2] R. Barzilay and M. Elhadad. Using lexical chains for text summarization. *Advances in automatic text summarization*, pages 111–121, 1999.

[3] S. O. CATEGORIZED. Keyword extraction based summarization of categorized kannada text documents. 2011.

[4] L.-F. Chien. Pat-tree-based keyword extraction for chinese information retrieval. In *ACM SIGIR Forum*, volume 31, pages 50–58. ACM, 1997.

[5] J. D. Cohen et al. Highlights: Language- and domain-independent automatic indexing terms for abstracting. *JASIS*, 46(3):162–174, 1995.

[6] J. M. Conroy and D. P. O'leary. Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 406–407. ACM, 2001.

[7] G. Ercan and I. Cicekli. Using lexical chains for keyword extraction. *Information Processing & Management*, 43(6):1705–1714, 2007.

[8] G. Erkan and D. R. Radev. Lexrank: graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, pages 457–479, 2004.

[9] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. Domain-specific keyphrase extraction. 1999.

[10] M. J. Giarlo. A comparative analysis of keyword extraction techniques. 2005.

[11] E. Hovy and C.-Y. Lin. Automated text summarization and the summarist system. In *Proceedings of a workshop on held at Baltimore, Maryland: October 13-15, 1998*, pages 197–214. Association for Computational Linguistics, 1998.

[12] A. Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223. Association for Computational Linguistics, 2003.

[13] J. K. Humphreys. Phraserate: An html keyphrase extractor. *Dept. of Computer Science, University of California, Riverside, California, USA, Tech. Rep*, 2002.

[14] S. Lappin and H. J. Leass. An algorithm for pronominal anaphora resolution. *Computational linguistics*, 20(4):535–561, 1994.

[15] M. Litvak and M. Last. Graph-based keyword extraction for single-document summarization. In *Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization*, pages 17–24. Association for Computational Linguistics, 2008.

[16] H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317, 1957.

[17] I. Mani and M. T. Maybury. *Advances in automatic text summarization*, volume 293. MIT Press, 1999.

[18] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):1993, pp. 313–330.

[19] Y. Matsuo and M. Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13(01):157–169, 2004.

[20] A. K. Mondal and D. K. Maji. Improved algorithms for keyword extraction and headline generation from unstructured text. *First Journal publication from SIMPLE groups, CLEAR Journal*, 2013.

[21] J. Ramos. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, 2003.

[22] G. Salton, A. Singhal, M. Mitra, and C. Buckley. Automatic text structuring and summarization. *Information Processing & Management*, 33(2):193–207, 1997.

[23] P. Turney. Learning to extract keyphrases from text. 1999.

[24] L. van der Plas, V. Pallotta, M. Rajman, and H. Ghorbel. Automatic keyword extraction from spoken text. a comparison of two lexical resources: the edr and wordnet. *arXiv preprint cs/0410062*, 2004.

[25] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. Kea: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*, pages 254–255. ACM, 1999.

[26] C. Zhang. Automatic keyword extraction from documents using conditional random fields. *Journal of Computational Information Systems*, 4(3):1169–1180, 2008.

[27] K. Zhang, H. Xu, J. Tang, and J. Li. Keyword extraction using support vector machine. In *Advances in Web-Age Information Management*, pages 85–96. Springer, 2006.