Text Summarization with Automatic *Keyword* Extraction in Telugu e-Newspapers

Reddy Naidu[1], Santosh Kumar Bharti[1], Korra Sathya Babu[1], and Ramesh Kumar Mohapatra[1]

[1]National Institute of Technology, Rourkela, Odisha, India - 769008
{naidureddy47,sbharti1984,prof.ksb}@gmail.com
{mohapatrark}@nitrkl.ac.in

Abstract. Summarization is the process of shortening a text document to make a summary that keeps the main points of the actual document. Extractive summarizers work on the given text to extract sentences that best express 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 work, we proposed an algorithm that automatically extracts keyword for text summarization in Telugu e-newspaper datasets. The proposed method compares with the experimental result of articles having the similar title in five different Telugu e-Newspapers to check the similarity and consistency in summarized results.

Keywords Automatic Keyword Extraction, e-Newspapers, NLP, Summarization, Telugu.

1 Introduction

Many popular Telugu e-newspapers are freely available on the internet, such as Eenadu, Sakshi, Andhrajyothy, Vaartha, Andhrabhoomi, etc. The extraction of all the relevant information from these newspapers is a tedious job for people. So, there is a need for a tool that extracts only relevant information from these data sources. To get the required information, we need to mine the text from newspapers. Natural Language Processing (NLP) is a powerful tool for text mining. Text Mining deploys some of the techniques of NLP and includes tasks like Automatic Keyword Extraction and Text Summarization. Summarization is a process where the most prominent features of a text are extracted and compiled into a short abstract of the original wording [1]. According to Mani and Maybury [2], 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% [3] of the original text and yet contain everything that could have been learned from reading the original article.

The Telugu language is the second most popular language in India just after Hindi, and it has got importance over other Indian languages as there are about 75 million native Telugu speakers. Telugu ranks fifteenth in the Ethnologue list

of most-spoken languages worldwide [4]. Telugu has rich agglutinative characteristics which motivated us to consider Telugu as the topic language over other Indian languages.

The rest of this paper is organized as follows: Related work is mentioned in Section 2. Proposed Scheme and implementation details of the paper are presented in Section 3. Experimental results are shown in Section 4. Finally, the conclusion of the paper presented in Section 5.

2 Related Work

In recent times, for the English language, many authors suggested the procedure for automatic keyword extraction in their state-of-the-art work [1], [5], [6]. Based on previous work done towards automatic keyword extraction from the text for its summarization, extraction techniques can be categorized into four approaches, namely, simple statistical approach, linguistics approach, machine learning approach, and hybrid approaches as discussed in subsequent sections. These are the approaches used for the Text Summarization in English language. In this paper, the proposed work follows a mixed approach of machine learning and statistical methods.

2.1 Simple Statistical Approach

These statistical methods are unprocessed, simplistic which do not require training data. They mainly 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. The methods of this approach include word frequency, term frequency (TF) or term frequency-inverse document frequency (TF-IDF), word co-occurrences and PAT-tree [5].

2.2 Linguistics Approach

This approach uses the linguistic features of the words in the sentences and articles. It includes the lexical analysis, syntactic analysis, discourse analysis, etc. Tree Tagger, WordNet, Electronic dictionary, N-grams, POS pattern, etc., are the primary resources of lexical analysis while Noun phrase (NP) chunks (Parsing) belong to syntactic analysis.

2.3 Machine Learning Approach

These approaches consider supervised or unsupervised learning from the examples, but previous work on keyword extraction prefer supervised method. 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.

2.4 Hybrid Approach

These approaches combine any of the above two mentioned methods or use heuristics, such as position, length, layout feature of the words, HTML tags [7] around the words, etc. These algorithms are designed to take the best features from above mentioned approaches.

3 Proposed Scheme

This section deals with explicit details on the approach that was used for automatic keyword extraction and summarization. The proposed algorithm follows a mixed approach of machine learning and statistical method. A Telugu POS Tagger [8] was used to identify appropriate POS information in Telugu text. Then, a statistical method is used to extract keywords. In this scheme, we are not considering the stop words to summarize the news. First, it extracts the keywords by the automatic keywords extraction algorithm and then to summarize the article, the keywords are determined to summarize the article. Finally, the summarization algorithm accordingly chooses sentences to form the necessitated summary. This algorithm applies to a single article at a time.

3.1 Automatic Keyword Extraction

The main 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 model is shown in Figure 1.

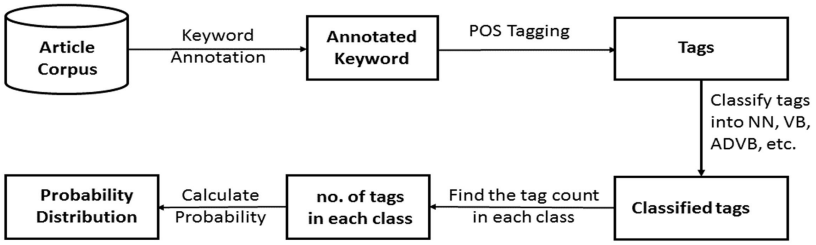

Fig. 1: Model for Learning of Probability Distribution

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.

Telugu POS Tagging: In this paper, we have used a Telugu POS tagger [8] 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). The Telugu POS tagger followed the Indian language standard tagset [9], which comprise 21 tags.

Learning Probability Distribution: Due to lack of reliable human annotators and it is a tedious job, 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 the number of nouns, verbs, adverbs, adjectives, etc. that appeared as a keyword was found in the headlines. The algorithm 1 is deployed for finding probability distribution values of keywords.

Algorithm 1: Find Probability Distribution

```
Data: dataset := dataset of Telugu e-news 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 newsarticle in dataset do
 | while keyword in newsarticle 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
```

Algorithm 1 infers P(tag) from the Telugu e-newspaper dataset. The value of the count variable is initialized to 0 which stores the number of keywords that has been scanned by the algorithm. It finds POS tag of the keyword for each keyword in every news article of the Telugu 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.

3.2 Extraction

Extraction (Testing) model is shown in Figure 2. The 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.

Keywords 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.* [Book, Noun] and [Book, Verb] are treated differently. When a count of the entire document is taken, the keywords are ranked by the Equation 1.

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

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.

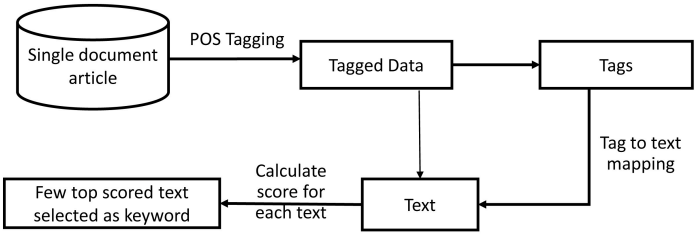

Fig. 2: Model for Keywords Extraction

Algorithm 2: Extract Keywords

Data: *doc* := Input Article
P(Tag) := List of Trained Probabilities
$Num\_Keywords$:= Required Number of Keywords
Result: $Keywords[]$
pos_doc:=pos_tagger(*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

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 a number of times that word-tag pair has been scanned. The input file 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 is then extracted as keywords.

3.3 Summarization

With the help of algorithms explained so far, a set of word tag pair keywords is attained and their respective scores. For summarization, the proposed algorithm suggests that one derives from many sentences for a keyword from the article as it 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.
To see the working procedure of the proposed scheme, let us assume a single document news article on Tamilnadu ex-CM Jayalalitha's (Amma) death. Possible keywords would be listed in Figure 3. Assume that one need to summarize it in twenty-five sentences. Given the individual scores of the keywords, as shown in Figure 3, we shall extract sentences for each of the keywords using Equation 2. Finally, the extracted number of sentences is shown in figure 3 to get the desired summary of the document.

$$NS = \lceil \frac{(Keyword\ score * No.of\ sentences\ required)}{(Total\ score of\ all\ the\ keywords)} \rceil \tag{2}$$

Where, NS = Number of Sentences needed in summary using each keyword.

Keyword	అమ్మ అస్తమయం	నింగికేగిన అమ్మ	మరణించిన అమ్మ	అమ్మ మహారాజ్ఞి ఇకలేరు
Keyword Score	3.5	2	3	1.5
NS	7	4	6	3

Fig. 3: Score of each keyword and Required number of sentences on each keyword

4 Results & Discussion

This section evaluates the quality of the keywords produced with the proposed algorithms. The section starts with article collection from different e-newspapers followed by results and discussion.

4.1 Article Collection

After analyzing the several e-newspapers of Telugu, we collected data from five different e-newspapers namely, Eenadu, Sakshi, Andhrajyothy, Vaartha, and Andhrabhoomi. Our dataset included almost 450 articles from each e-newspapers ranging from the 1st of October 2016 to 6th of December 2016. We have collected 150 articles with a total of 1223 keywords, 140 articles with a total of 1148 keywords, 100 articles with a total of 915 keywords, 70 articles with a total of 785 keywords, 50 articles with a total of 568 keywords from Eenadu, Sakshi, Andhrajyothy, Vaartha and Andhrabhoomi e-newspapers respectively.

4.2 Experimental Results

In this paper, work has experimented in a machine with the following configuration:

- *System Configuration* : Intel(R) core(TM) i7-4770 CPU @ 3.40 GHz with 4 GB RAM and minimum 20 GB memory space.
- *Operating System used* : Windows 7 Professional X 32
- *Software Package used* : In Windows, Python Interpreter.

Table 1: Number of times a tag has been found in different e-Newspapers headlines and their probability measures

Tag Count	**Newspaper**	**VM**	**NN**	**PRP**	**RB**	**INJ**	**JJ**
	Eenadu	350	245	255	152	154	76
	Sakshi	317	185	250	195	78	85
	Andhrajyothy	278	195	142	105	47	73
	Vaartha	207	143	103	168	65	95
	Andhrabhoomi	177	125	93	75	44	55
Probability Measures							
	Eenadu	0.286	0.200	0.208	0.124	0.126	0.062
	Sakshi	0.275	0.161	0.217	0.169	0.067	0.074
	Andhrajyothy	0.303	0.213	0.155	0.114	0.051	0.079
	Vaartha	0.263	0.182	0.131	0.214	0.082	0.121
	Andhrabhoomi	0.311	0.220	0.163	0.132	0.077	0.096

To evaluate the performance of proposed algorithm and compare the results with existing work, three parameters are considered, namely, *precision*, *recall* and $f - score$.

Precision is a measure of result relevancy, while *Recall* is a measure of how many truly relevant results are returned. $Precision(P)$ is defined as the number of true positives over the number of true positives (T_p) plus the number of false positives (F_p). $Recall(R)$ is defined as the number of true positives (T_p) over the number of true positives (T_p) plus the number of false negatives (F_n).

For testing, we ran our algorithm on a set of Telugu e-newspaper articles from the above mentioned five e-newspapers. However, this time, the headlines were not provided to the algorithm. We collected the content of the article with similar article title in all five e-newspapers (same news in all five 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. The result of Table 2 shows how effectively our proposed algorithm work on all five e-newspapers, and all of them attains almost similar *accuracy*, *precision*, *recall* and $f - score$ values. As of my knowledge,

there is no reported work for Text Summarization in Telugu Language. So, the results are not compared to any of the work.

Table 2: Results interms of Confusion Matrix, *Accuracy*, *Precision*, *Recall*, $F-score$

Newspaper	T_p	F_p	T_n	F_n	*Accuracy*	*Precision*	*Recall*	$F-score$
Eenadu	198	51	912	62	0.907	0.795	0.761	0.777
Sakshi	165	41	887	54	0.917	0.800	0.753	0.776
Andhrajyothy	183	35	653	44	0.913	0.839	0.806	0.822
Vaartha	174	27	545	39	0.915	0.865	0.816	0.840
Andhrabhoomi	153	27	352	36	0.889	0.85	0.809	0.829

5 Conclusion

In this paper, the proposed work dealt with interdependent algorithms in keyword extraction and text summarization. The keyword extraction algorithm found the top scored keywords as efficiently as human do. With the help of keyword extraction algorithm, a summarization algorithm was proposed that introduced a concept that primary objective is "not all keywords are equal".

References

1. Litvak M. and Last M.: Graph-based keyword extraction for single-document summarization. In Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization, pp. 17–24, ACL (2008)
2. Mani I. and Maybury M.T.: Advances in automatic text summarization, volume 293, MIT Press (1999)
3. Thomas J.R., Bharti S.K., and Babu K.S.: Automatic Keyword Extraction for Text Summarization in e-Newspapers. In Proceedings of the International Conference on Informatics and Analytics, pp. 86–93, ACM (2016)
4. http://www.ethnologue.com/statistics/size.
5. Chien L.F.: Pat-tree-based keyword extraction for chinese information retrieval. In ACM SIGIR Forum, Vol. 31, pp. 50–58, ACM (1997)
6. Giarlo M.J.: A comparative analysis of keyword extraction techniques (2005)
7. Humphreys J. K.: An html keyphrase extractor. Dept. of Computer Science, University of Cal-ifornia, Riverside, California, USA, Technical Report (2002)
8. Reddy S. and Sharoff S.: Cross Language POS Taggers (and other Tools) for Indian Languages An Experiment with Kannada using Telugu Resources. In Proceedings of IJCNLP workshop on Cross Lingual Information Access: Computational Linguistics and the Information Need of Multilingual Societies. Chiang Mai, Thailand (2011)
9. Bharati A., Sangal R., Sharma D.M. and Bai L.: Anncorra: Annotating corpora guidelines for pos and chunk annotation for indian languages. Technical Report. Technical Report (TRLTRC-31), LTRC, IIIT-Hyderabad (2006)