VERSITA

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **12**, No 2

Sofia • 2012

Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2012-0011

Summarizing News Paper Articles: Experiments with Ontology-Based, Customized, Extractive Text Summary and Word Scoring

Jagadish S. Kallimani, K. G. Srinivasa**, Eswara Reddy B.****

**Research Scholar, Department of Computer Science and Engineering, Jawaharlal Nehru Technological University, Kakinada, Andra Pradesh, India*
***Department of Computer Science and Engineering, M S Ramaiah Institute of Technology, Bangalore, India*
****Department of Computer Science and Engineering, Jawaharlal Nehru Technological University, Anantapur, Andra Pradesh, India*
*Emails: **j**sk_msrit@rediffmail.com srinivasa.kg@gmail.com eswarcsejntu@gmail.com*

Abstract: *The method for filtering information from large volumes of text is called Information Extraction. It is a limited task than understanding the full text. In full text understanding, we express in an explicit fashion about all the information in a given text. But, in Information Extraction, we delimit in advance, as part of the specification of the task and the semantic range of the result. Only extractive summarization method is considered and developed for the study. In this article a model for summarization from large documents using a novel approach has been proposed by considering one of the South Indian regional languages* (*Kannada*)*. It deals with a single document summarization based on statistical approach. The purpose of summary of an article is to facilitate the quick and accurate identification of the topic of the published document. The objective is to save prospective readers' time and effort in finding the useful information in a given huge article. Various analyses of results were also discussed by comparing it with the English language.*

Keywords: *Information extraction, extractive summarization, automatic text summarization, text summarization, stemming, word count frequency, UTF-8.*

1. Introduction

Natural Language Processing (NLP) is the engineering of systems that process or analyze written or spoken natural language. It is a field in artificial intelligence which attempts to use computers to process information contained in ordinary language such as English. The main problem with NLP is the acquisition of a large amount of information of the world [1]. The information extraction includes the retrieval of documents from collections and the tagging of particular terms in text. So it can be defined as the identification of instances of a particular class of events or relationships in a natural language text, and the extraction of the relevant arguments of the event or relationship. Information extraction involves the creation of a structured representation such as a data base of the selected information drawn from the text.

Text summarization is a way to condense the large amount of information into a concise form by the process of selection of important information and discarding unimportant and redundant information. With the amount of textual information present in the World Wide Web (www), the area of Automatic Text Summarization (ATS) is becoming very important in the field of information retrieval. Some applications that text summarization can support are: news summarization down to SMS or WAP for mobile phones, make computers read a summarized text, written text can be too long to read, to present short description of a matching text by the search engines and in a foreign language, to obtain a short translated text of a summarized text.

There has been a rapid increase in the amount of researches done in the field of automatic text summarization. Text summarization is also increasingly being exploited in the commercial sector, in the telecommunications industry (British Telecommunication's ProSum), data mining of text databases (Oracle's Context), in filters for web based information retrieval (Inxight's summarizer used in AltaVista Directory). In addition to the traditional focus of automatic indexing and automatic abstracting to support information retrieval, researchers are investigating the challenging problems, including multilingual summarization. As the information overload problem grows, the information becomes more and more accessible (via mobile devices, etc.). The field of automatic text summarization can be expected to become more and more important, and the new applications for the text summarization will surface.

The search engines do a remarkable job in searching through a heap of information to dish out the most relevant information that the user is looking for. It is huge time consuming to read through the entire length of the document. Invariably a decision for a certain task has to be made in a certain time frame and reading through all the documents is simply impossible. The process speeds up considerably when the essence, i.e., the summary of the document is also available. The technology of automatic text summarization is critical when dealing with such situations.

2. Related works

Types of automatic text summarization

Automatic text summarization can be generated using different concepts or methods that are based on the type of the article to be summarized and the user requirements. Each method has got advantages and disadvantages which are discussed in detail.

2.1. Generic and topic centric summarization

Generic summaries are the summaries that give users the overall sense of the document. Most of the summaries created by the human being are generic summaries. A generic summary typically must contain the core information present in the document. A document normally consists of sentences that give information to the user about a single broad or narrow matter, but invariably the understanding of the matter requires bits and pieces of information from other domains too. One of the most notable approaches to generic summarization has been done by Carbonell and Goldstein, based on Maximal Marginal Relevance (MMR). MMR uses the vector-space model of text retrieval and is particularly applicable to query-based and multi-document summarization.

In case of topic centric summarization, the main task is to identify the topics present in the document. Identification of topics is important even in a single document summarization. In real life applications, the main purpose of automatic text summarization for single documents is to summarize news articles. Although the news articles concentrate on one central topic, they also contain information other than the central one. Once the topics have been identified, the sentences from each of the topics can be used to generate the summary. Probabilistic Latent Semantic Indexing (PLSI) has been used as the main method to identify the topics. The topic centric summaries are generally restricted to one topic. The topic is based on the key word, supplied typically by humans. The summary may be created from a single or multiple documents. This kind of summary has a great potential in future search engines.

2.2. Abstractive and extractive summarization

The abstractive summarization is the process in which the abstract of the document is created. The abstract can contain the words and phrases, which may not be present in the original document. The extractive summarization is the one, where the exact sentences present in the document are used as summaries. The abstractive summarization procedure is a very complicated process as the semantics of sentences has to be dealt with. Several other factors such as word sense, grammatical structure has to be taken into consideration before creating a useful abstractive summary. The abstractive summary that has all the characteristics of a good summary is the ultimate goal of automatic text summarization. Abstractive summary is important because the textual structure present in the summary can vary significantly from the original text. The copyright of the original text would not be infringed and the intellectual rights for the summary can be owned by the summarizing party. The extractive summarization is simpler than abstractive

summarization and is the general practice among the automatic text summarization researchers at the present time. Extractive summarization process involves in giving scores to sentences using some naval approaches and then generating the summary by using the sentences that have the highest scores as summaries. This kind of a summary is generally completely unsupervised and language independent too.

2.3. Single and multi-document summarization

Single document summarization is, as the name suggests, the process of summarizing one document. Whereas multi-document summarization is where multiple documents (related to one main topic) are used as sources of the summary. The resulting represents the core information present in all of the documents. Both kinds of summaries have their own advantages. Single document summarization is a more traditional way of creating a summary. It is useful in most of the situations, such as summarizing emails, news or creating abstract of scientific research papers etc.

With the amount of information present in Internet, the technology of multi document summarization is becoming more and more important. For example, we can get news related to a single event or a person from several sources. If we use all the sources to create the summary in an efficient manner, we can get a summary, where each topic is described from multiple perspectives. Multi-document summarization is a very challenging field. The major challenge arises due to the presence of diverse themes in a set of documents. The main goal of multi-document summarization is to combine the main themes with completeness, readability and conciseness. An ideal multi-document summarization system does not simply shorten the source text but also presents information organized around the key aspects to represent a wider diversity of views on the topic. When such quality is achieved, an automatic multi-document summary is perceived more like an overview of a given topic. Multi-document summarization poses interesting challenges beyond single documents. An important study shows that for the newspaper article genre, even some very simple procedures provide essentially perfect results. For example, taking the first two or three paragraphs of the most recent text of a series of texts about an event provides a summary which is equally coherent and complete as that produced by human abstracters. Obviously, this cannot be true for more complex types of a summary, such as biographies of people or descriptions of objects. Further research is required on all aspects of multi-document summarization before it can become a practical reality.

Automatic indexing research [2, 3] in the 1970-ies and 1980-ies evolved into statistical processing methods based on *tf.idf* weighting, which applies significance to a term by counting the number of times it appears in a document (the *term frequency)* and multiplying the result by the term's *inverse document frequency, idf)* – a logarithmic calculation of the total number of documents in the collection, divided by the number of documents containing the target term. In the 1990-ies, Salton used *tf.idf* weights and other measures derived from indexing research to identify closely related segments within a document and then to compare relationships with those of other documents, generating automatic hyperlinks when the similarities were close.

The *summarist* was developed at the University of Southern California; Institute of Information Sciences (ISI) [4].The prime target of the *summarist* is to provide a good summarization, based on the following formula:

Summarization = topic identification + interpretation + generation

Both extract and abstract summaries can be generated by this system. Each step contains modules trained on a large corpora of text. The first stage filters the given text to determine the most important topics. This technique combines NLP methods using statistical techniques (extract) with symbolic word knowledge (abstract) provided by dictionaries, *WordNet* and other resources.

SweSum [5, 6] is a web- based text summarizer, developed at KTH (Kungliga Teckniska Högskolan), Stockholm. Extraction methods are being used which are based on statistical, linguistic and heuristic methods. It is all implemented in Perl language and the domain is HTML-tagged newspaper text. Besides Swedish language [7, 8], the latest versions include Norwegian, Danish, Spanish, French, German and a version for Farsi language (Iranian) [9]. It has been evaluated for Swedish, Danish, Norwegian, French and Farsi with the results being very encouraging. The French, German and Spanish versions are in prototype states [10]. Although extract summarization is easy to implement, there are three major difficulties: finding out which are the most important sentences to use for the summary? How to generate a coherent summary? and How to remove all redundancies in the summary?

The scientific society has proposed some solutions to these problems. To handle the sentence choosing, a method of scoring the candidate sentences has been developed. Among those the ones with the highest scores are chosen for the final summary. Some rules according to which sentences are rated are presented below [11]:

- *Baseline:* This is a scoring system according to which the sentences take their marks depending on their place on the text. For newspaper texts, the first sentence of the text gets the highest ranking, while the last get the lowest.
- *First sentence:* Similarly to the previous condition, the first sentence of each paragraph of the text is considered to be very important.
- *Title:* The words included in the title along with the following sentences get a high score.
- *Word frequency:* Words, called open class words, which are frequent in the text, are more important than less frequent. The sentences including such keywords that are most often used in the passage usually represent the topic of it.
- *Indicative phrases:* Sentences containing phrases like "…this document…"
- *Position score:* There is a theory that certain types of documents have their key meaning in certain parts of it. For example, in the newspaper text, the first four paragraphs are the most important, while in technical papers the conclusion section is the most important part.
- *Sentence length:* The score given to a sentence reflects the number of words in a sentence, normalized by the length of the longest text in the passage.
- *Proper name:* Sentences which contain proper nouns get a higher scoring.

- *Average lexical connectivity:* The sentences that share more terms with other sentences are scored higher.
- *Numerical data:* The sentences that contain any sort of numerical data are scored higher than those that do not contain.
- *Proper name:* Certain types of nouns, like people's names, cities, places, etc. are important in newspaper texts and sentences containing them are scored higher.
- *Pronoun:* Sentences containing a pronoun (reflecting co-reference connectivity) are scored higher than those that do not contain.
- *Weekdays and months:* Sentences containing names of weekdays or months are scored higher.
- *Quotation:* Sentences containing quotations may be important for some sort of questions, which are the input by the user.
- *Query signature:* When a user requires a summary he/she usually has a certain topic on his/her mind. The query of the user affects the summary in that the extracted text will be compelled to contain these words. Normalized score is given to sentences depending on the number of query words that they contain.

The rules described above are very important to text summarization, but alone are not enough to produce a good quality summary. Additionally, several word-level techniques, such as synonymy, polysemy and phrases [6] also influence the summarization process. In the proposed work of summarization for Kannada language, the first sentence, word frequency, position score and quotation are considered.

3. The proposed work

Keeping the above facts in view, the present study deals with the following objectives:

- To understand the general concepts of text summarization tool and develop it.
- To study the impact of stemming in the text summarization process.
- To carry out Kannada language text summarization using the well established extractive summarization method, by developing and incorporating the Kannada dictionary relevant to the given article.
- Performance studies on text summarization on both English and Kannada languages.

3.1. Kannada language and UTF-8

Kannada is a south Indian language spoken in Karnataka state of India. Kannada has originated from the Dravidian language. Telugu, Tamil, Malayalam are the other South Indian languages originated from Dravidian language. Kannada and Telugu have almost the same script. Malayalam and Tamil have resemblance. Kannada as a language has undergone modifications since BCs. It can be classified into four types:

Purva Halegannada (from the beginning till 10th Century)

Halegannada (from 10th Century to 12th Century)
Nadugannada (from 12th Century to 15th Century)
Hosagannada (from 15th Century)

Kannada is a verb-final inflectional language with a relatively free word order. Kannada morphology is characterized as agglutinative or concatinative, i.e., words are formed by adding suffixes to the root word in a series. Most of the words may change spelling when stems are inflected. Normally the root word is affixed with several morphemes to generate thousands of word forms. The complexity of developing morphological analyzer and generator for Dravidian language like Kannada is comparatively higher than for other languages like English. Most of the words may change spelling when stems are inflected. In agglutinative language like Kannada normally a root word is affixed with several morphemes to generate thousands of word forms. To build an effective morphological analyzer one should carefully analyze and identify all these roots and morphemes. Due to the highly agglutinating nature of the Kannada language and the morphophonemic variations that take place at the point of agglutination, it is very difficult to mark word boundaries [12]. Design should possibly cover all types of inflections.

The Kannada block of Unicode standard (0C80 to 0CFF) is based on ISCII-1988 (Indian Standard Code for Information Interchange). The Unicode standard (Version 3) encodes Kannada characters in the same relative positions as those coded in the ISCII-1988 standard. UTF-8 (***Unicode Transformation Format, 8-bit encoding***) is a method for encoding Unicode text. Unicode is fast becoming the standard for encoding characters from ***all*** written languages. In a nutshell it is simply an agreement by all parties that any given character from any given language will ***always*** be represented by the same numeric value. ***Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language.***

3.2. The AutoSum

The *AutoSum* is a text summarizer. The program reads a text and decides which sentences are important and which are not. It will create a short summary or will highlight the main ideas in the text. The interaction with *AutoSum* is through a command line. It lets us to summarize a text on the console. The program can either print the summarized text as a text or HTML. If in HTML, the important sentences are highlighted. The program works with UTF-8 encoding.

The summarization process starts when the user enters text, which involves the steps as shown in Fig. 1 [13]. The user gives commands on the terminal. In several steps, the Kannada text gets through the program and is being summarized. After the process of summarization, the resulting summarized Kannada text is returned to the terminal or the output summary can be pipelined to some other new text file or the summarized result can be viewed in the web browser in a highlighted form.

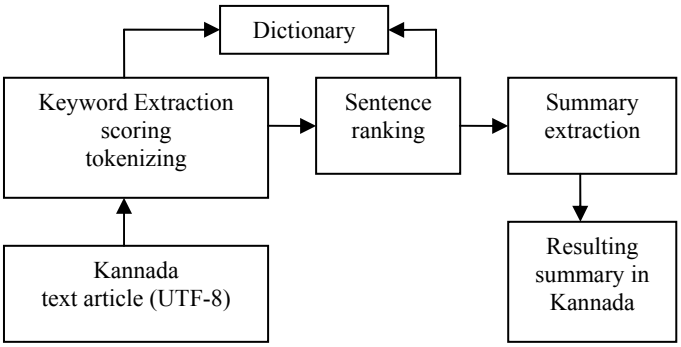

Fig. 1. The processes involved in *AutoSum*

Keyword extractions are essential for text summarization, but there is a variety of different types used by the research community. *AutoSum* uses nouns, adjectives and adverbs as keywords. In order to extract the keywords in our system, two different methods have been developed: tagging and parsing of text and using a lexicon. Using a lexicon has been proven to be a lot easier and much faster. In *AutoSum* [14, 15] the keyword frequency counting is based on the types of keywords described previously, by using a static lexicon. When the program is asking the lexicon for a word, the second returns the lemma of the word that is stored in a word frequency hash table. The same words with different inflections are not being counted as different words.

In order to determine the importance of every sentence on the text, a special scoring system has to be introduced. The sentences that contain no text are not summarized and are labeled *"not text."* Those that contain the text to be summarized are labeled with the value *"text"*. The lines that are marked as *"text"* are put in a data structure for storing key/value, which is named Text Table Value. The line content is the key to the table and the line number and value is the score of the line. This depends on the position of the line and also how the words in this line are scored. In the final summary, only those with high scores are displayed.

In *AutoSum* [16, 17] the following is used for calculating the line score:

- First line: It should always be included in the summary. This is done by assigning it a very high score.
- Position: The position score depends on the text to be summarized. As mentioned before, in newspaper text the first line is the most important and gets the highest score, followed by the others. There is a mathematical formula being used for defining the position score

$$\text{Position score} = (1/\text{line number}) \times 10.$$

- Numerical values: Whenever a number is identified in a text, the line that includes it gets one additional point.
- Keywords: They are automatically identified as the most frequent words in the passage. The sentences that contain more keywords take a higher score than those that contain fewer or none.
- Simple combination function: The different methods described above are combined and used to calculate the score of each sentence.

The score of every word is being identified and added to the sentence score in the text table value.

Sentence scoring

Word score = (word frequency)

Sentence Score = Σ Word Score

For Example, in a considered article:

Bush has little choice but to compromise on the tax cuts

(in the given inputted text)

Word score (Bush) = 16

Word score (little) = 3

Word score (tax) = 16

Word Score (cuts) = 11

Sentence score = Word score (Bush) + Word score (little) + Word score (tax) + Word score (cuts) = 46

Cutoff size and unit: **Along with text insertion, the user has the option to insert two kinds of values, cutoff size and unit. The cutoff size is a numerical value and unit is either percent, number of words or characters.**

User Input: Cutoff size = 20, unit = percent.

Summary Result: Keeps 20% of the words of the original text

Important ideas in an article are described with many of the same words while redundant information uses less technical terms and is not related to the main subject of the article. Important lines are the lines that are related to the subject of the article. The subject of the article is scanned once and the words and their occurrence in the text are stored in a list. This list will be sorted by the occurrence of words in the text. Now it will remove all the words that are common in the language using a dictionary file. These words are the words that do not teach us anything about the subject of the article. Knowing that the word 'tax' in a given text can tell us that the article might talk about 'tax', however knowing that the word 'of' in the text can not teach us anything about the subject of the text. After removing the redundant words, a new list will be generated. From this list we may assume that the text tells us about "president, Bush, tax, cuts, economy". So an important sentence in the text will be a sentence that talks about "president, Bush, tax, cuts and economy". A sentence that talks about France or Paris may be neglected as they appear only once in the text. So we can assume that France or Paris is not one of the main ideas in the text. Each sentence is given a grade based on the keywords in it. A line that holds many important words will be given a high grade. To produce a 20% summary it prints the top 20% sentences with the highest grade. The process of summary generation generally has three major constituents: key word selection, sentence weighting, and sentence selection.

Selection of key words: It utilizes the individual word statistics to determine the sentences of interest. During extract generation, it selects sentences for the summary by utilizing a list of key words. The individual word weights are calculated by employing a term frequency method. This extended list of key words represents words that are likely to indicate a topic or convey other important information. There is no limit on the number of key words used for a given

document. All topical key words exceeding an empirically determined significance threshold and all infrequent headline words are included [18].

Sentence weighting and selection: Sentence selection chooses the sentences that will become the final document summary. Sentence weighting determines which sentences contain information relating to the main ideas in the document based on the previously identified key word list. Each sentence's weight is computed by summing the weights of the individual key words present in the sentence; the sentence weights are used to choose the summary sentences. For each document a set of sentences is chosen based on a number of factors, such as: the presence of key words in the sentence; its location in the document; the target percentage of the extract.

3.3. Stemming

Removing suffixes [19] by automatic means is an operation which is especially useful in the field of information retrieval. In a typical information retrieval environment, one has a collection of documents, each described by the words in the document title and possibly by words in the document abstract. Ignoring the issue of precisely where the words originate, we can say that a document is represented by a vector of words or terms. Terms with a common stem will usually have similar meanings, for example:

CONNECT
CONNECTED
CONNECTING
CONNECTION
CONNECTIONS

Frequently, the performance of an information retrieval system will be improved if term groups, such as this are conflated into a single term. This may be done by removal of the various suffixes -ED, -ING, -ION, -IONS to leave the single term CONNECT. In addition, the suffix stripping process will reduce the total number of terms in the system, and hence reduce the size and complexity of the data in the system, which is always advantageous.

Many strategies for suffix stripping have been reported in literature. The nature of the task will vary considerably depending on whether a stem dictionary is being used, whether a suffix list is being used, and of course, on the purpose for which the suffix stripping is being done. Assuming that one is not making use of a stem dictionary, and that the purpose of the task is to improve information extraction performance, the suffix stripping program will usually be given an explicit list of suffixes and with each suffix, the criterion under which it may be removed from a word to leave a valid stem. This is the approach adopted here. In any suffix stripping program, two points must be considered. Firstly, the suffixes are being removed simply to improve IE performance, and not as a linguistic exercise. This means that it would not be at all obvious under what circumstances a suffix should be removed, even if we could exactly determine the suffixes of a word by automatic means.

Perhaps the best criterion for removing suffixes from two words W1 and W2 to produce a single stem S, is to say that we do so if there appears to be no difference between the two statements "a document is about W1" and "a document is about W2". So if W1= "CONNECTION" and W2= "CONNECTIONS" it seems very reasonable to conflate them to a single stem. But if W1= "RELATE" and W2= "RELATIVITY" it seems perhaps unreasonable, especially if the document collection is concerned with theoretical physics. (It should perhaps be added that RELATE and RELATIVITY are conflated together) Between these two extremes there is a continuum of different cases, and given two terms W1 and W2, there will be some variation in opinion as to whether they should be conflated, just as there is with deciding the relevance of some document to a query.

The second point is that the use of a suffix list with various rules, the success rate for the suffix stripping will be significantly less than 100 % irrespective of how the process is evaluated. For example, if SAND and SANDER get conflated, so most probably will WAND and WANDER. The error here is that the -ER of WANDER has been treated as a suffix when in fact it is a part of the stem. Equally, a suffix may completely alter the meaning of a word, in which case its removal is unhelpful. PROBE and PROBATE for example, have quite distinct meanings in modern English. As said earlier, Kannada morphology is characterized as agglutinative or concatinative, i.e., the words are formed by adding *suffixes* to the root word in a series. Most of the words may change spelling when *stems* are inflected. Normally a root word is affixed with several morphemes to generate thousands of word forms. In agglutinative language like Kannada normally a root word is affixed with several morphemes to generate thousands of word forms. To build an effective morphological analyzer one should carefully analyze and identify all these roots and morphemes.

The algorithm. To present the suffix stripping algorithm in its entirety, few definitions [30] are needed. A consonant will be denoted by *c*, a vowel by *v*. A list *ccc*... of length greater than 0 will be denoted by C, and a list *vvv*... of length greater than 0 will be denoted by V. Any word, or part of a word, therefore, has one of the four forms:

CVCV ... C
CVCV ... V
VCVC ... C
VCVC ... V

These may all be represented by the single form [C] VCVC ... [V] where the square brackets denote arbitrary presence of their contents. Using (VC) {*m*} to denote VC repeated *m* times, this may again be written as [C](VC){*m*}[V]. *m* will be called the *measure* of any word or word part when represented in this form. The case *m* = 0 covers the null word. Here are some examples in Table 1.

Table 1

Measure	Word/Word part
m=0	TR, EE, TREE, Y, BY
m=1	TROUBLE, OATS, TREES, IVY
m=2	TROUBLES, PRIVATE, OATEN, ORRERY

The *rules* for removing a suffix will be given in the form (condition) S1 → S2.

This means that if a word ends with the suffix S1 and the stem before S1 satisfies the given condition, S1 is replaced by S2. The condition is usually given in terms of *m* for instance (*m* > 1) EMENT. Here S1 is "EMENT" and S2 is null. This would map REPLACEMENT to REPLAC, since REPLAC is a word part for which *m* = 2. The content of the "condition" part may be also as it is shown in Table 2.

Table 2

*S	The stem ends with S (and similarly for the other letters)
v	The stem contains a vowel
*d	The stem ends with a double consonant (e.g. -TT, -SS)
*o	The stem ends *cvc*, where the second c is not W, X or Y (e.g. -WIL, -HOP)

The condition part may also contain expressions with \and\, \or\ and \not\, so that (*m*>1 and (*S or *T)) tests for a stem with m>1 ending in S or T, while (*d and not (*L or *S or *Z)) tests for a stem ending with a double consonant other than L, S or Z. Elaborate conditions like this are required only rarely. In a set of rules written beneath each other, only one is obeyed, and this will be the one with the longest matching S1 for the given word. For example, with

SSES → SS
IES → I
SS → SS
S → (here the conditions are all null) CARESSES maps to CARESS since SSES is the longest match for S1. Equally CARESS maps to CARESS (S1= "SS") and CARES to CARE (S1= "S").

In the rules below, examples of their application, successful or otherwise, are given on the right in lower case.

The suffix stripping algorithm

Step 1a

SSES → SS	caresses → caress
IES → I	ponies → pony
SS → SS	caress → caress
S →	cats → cat

Step 1b

(m>0) EED → EE	feed → feed
(*v*) ED →	plastered → plaster
(*v*) ING →	motoring → motor

If the second or third of the rules in Step 1b is successful, the following is done:

AT → ATE	conflat(ed) → conflate
BL → BLE	troubl(ed) → trouble
IZ → IZE	siz(ed) → size
(*d and not (*L or *S or *Z)) → single letter	
	hopp(ing) → hop

tann(ed) → tan
fall(ing) → fall
fail(ing) → fail
fil(ing) → file

The rule to map to a single letter causes the removal of one of the double letter pair. The -E is put back on -AT, -BL and -IZ, so that the suffixes -ATE, -BLE and -IZE can be recognized later. This E may be removed in Step 4.

Step 1c

(*v*) Y → I happy → happi
sky → sky

Step 1 deal with plurals and past participles. The subsequent steps are much more straightforward.

Step 2

(m>0) ATIONAL → ATE relational → relate
(m>0) TIONAL → TION conditional → condition
(m>0) ENCI → ENCE valenci → valence
(m>0) ANCI → ANCE hesitanci → hesitance
(m>0) IZER → IZE digitizer → digitize
(m>0) ABLI → ABLE conformabli → conformable
(m>0) BILITI → BLE sensibiliti → sensible

The test for the string S1 can be made fast by doing a program switch on the penultimate letter of the word being tested. This gives a fairly even breakdown of the possible values of the string S1. It will be seen in fact that the S1-strings in step 2 are presented here in the alphabetical order of their penultimate letter. Similar techniques may be applied in the other steps.

Step 3

(m>0) ICATE → IC triplicate → triplic
(m>0) ATIVE → formative → form
(m>0) ALIZE → AL formalize → formal

Step 4

(m>1) AL → revival → reviv
(m>1) ANCE → allowance → allow
(m>1) ENCE → inference → infer
(m>1) ER → airliner → airlin
(m>1) IC → gyroscopic → gyroscop

The suffixes are now removed.

Step 5a

(m>1) E → probate → probat
(m=1 and not *o) E → cease → ceas

Step 5b

(m > 1 and *d and *L) → single letter

The algorithm is careful not to remove a suffix when the stem is too short, the length of the stem being given by its measure *m*. There is no linguistic basis for this approach. It was merely observed that *m* could be used quite efficintly to help decide whether or not it was wise to take off a suffix. Two lists are given for example in Table 3.

Table 3

List A	List B
RELATE	DERIVATE
PROBATE	ACTIVATE

ATE is removed from the list B words, but not from the list A words. This means that the pairs DERIVATE/DERIVE, ACTIVATE/ACTIVE will conflate together.

On the principle of *AutoSum*, an attempt has been made to summarize the Kannada articles, for which a dictionary relevant to the given article was developed in XML and incorporated into *AutoSum*. The common non-informative words are eliminated. All the parameters, such as percentage summarization, key words of the article and term count of key words were tried.

4. Performance studies

This section shows details about the text summarizer, how it displays or presents the results when an article is given as input. The summarizer will be shown different screens based on the type of commands and requirements. This customized way of generation of summary makes our work unique, reliable and user friendly. The required and customized percentage of summary (such as 20 %, 30 %) in both HTML format and text format, listing of keywords encountered in the text document and term count are obtained. The term count in the article can be displayed with and without stemming of words. Keeping the above facts in view, a Kannada dictionary relevant to the given article was developed in XML format. Using this dictionary, the above types of results were obtained.

Analysis of results are given in the comparative analysis between a machine and human generated summary.

A comparative analysis of 20 % summary generated using the sample article by a machine with human was conducted. The article was given to five different individuals and they were asked to generate a 20 % summary, the average of those five summaries ware generated by taking the common sentences of each summary and that average summary was used as a human summary for comparison with a machine generated summary. The following results were obtained for both English and Kannada languages.

- Total word count at 20 %, 30 % and 40 % summary size.
- Common and uncommon sentences generated in the summaries of both languages.

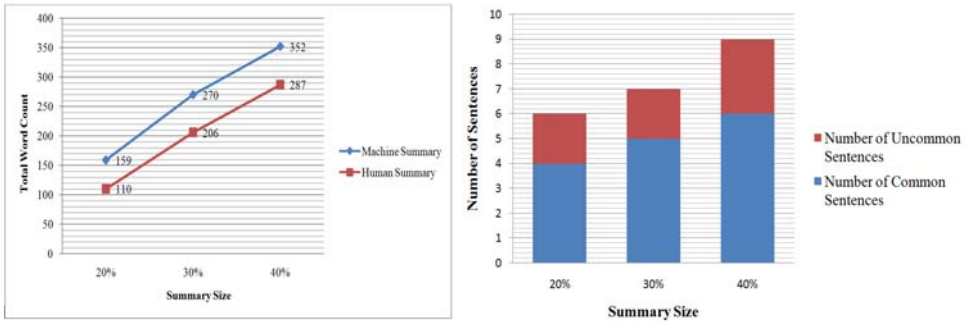

Fig. 2. The total word count and common sentences for English language article

It can be interpreted from Fig. 2 that the total numbers of words in a machine generated summary are more compared to the human generated summary and the number of the word count has increased as the summary size is increased in both cases indicating the efficient and comparable summarization process. In the 20 % summary out of the six sentences, four (66.66 %) are common, in the 30 % summary out of the seven sentences, five (71.42 %) are common and in the 40 % summary out of the nine sentences, six (66.66 %) are common. It indicates that a minimum of 66.66 % sentences are common in both human and machine generated summaries.

Similarly, comparative analysis was made for Kannada language article as follows:

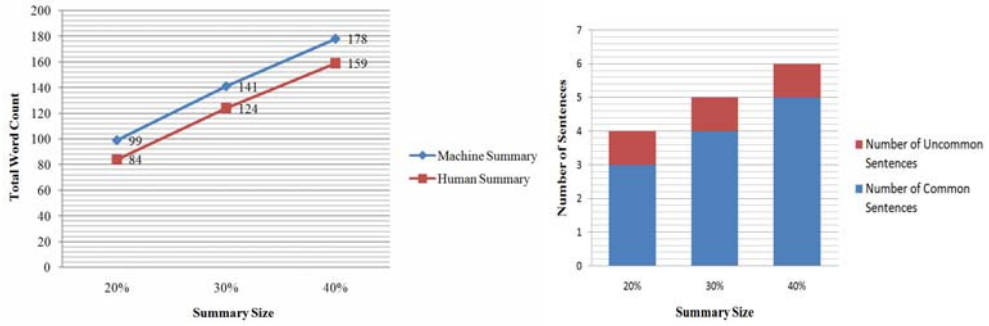

Fig. 3. The total word count and common sentences for Kannada language article

Based on Fig. 3, it can be interpreted that in Kannada summarization similar to English, the total number of word count in a machine generated summary are more compared to the human generated summary and the percentage of the word count has increased in both summaries as the summary size is increased. It is evident that in the 20 % summary out of the four sentences, three (75 %) are common, in the 30 % summary out of the five sentences, four (80 %) are common and in the 40 % summary out of the six sentences, five (83.33 %) are common. It indicates that as the percentage of the summary size is increasing, the percentage of the common sentences has also increased.

5. Conclusions

In this article the process of text summarization has been discussed. A Kannada lexical database in XML format was also developed relevant to the given article and used to summarize (*KanSum*) the article using the concept of *AutoSum*. Different analyses were carried out like 20 %, 30 % summarization and keyword extraction. The results have clearly shown that the system (*KanSum*) is working efficiently on par with *AutoSum* concept. In future a large scale Kannada lexical data base can be created in XML format as other languages and efficiently utilize *KanSum* on a large scale to produce the summaries of Kannada articles. As in English language with different summarization tools, *KanSum* can also be used to extract words from multiple sentences and the huge data can be reduced/summarized as per the requirements.

References

1. Kallimani, J. S., K. G. Srinivasa, R. B. Eswara. Information Extraction by an Abstractive Text Summarization for an Indian Regional Language. – In: Proceedings of IEEE NLP-KE, Tokushima, Japan, 27-29 November 2011.
2. Salton, G. Another Look at Automatic Text Retrieval Systems. – Communications of the ACM, Vol. **29**, July 1986, No 7, 648-656.
3. Salton, G., C. T. Yang. A Theory of Term Importance in Automatic Text Analysis. – Journal of the ASIS, Vol. **26**, January-February 1975, No 1, 133-44.
4. Lin, C. Y, E. Hovy. Identify Topics by Position. – In: Proceedings of the 5th Conference on Applied Natural Language Processing, March 1997.
5. Dalianis, H. SweSum – A Text Summarizer for Swedish. TRITA-NA-P0015, IPLab-174, NADA, KTH, October 2000.
   http://www.dsv.su.se/~hercules/papers/Textsumsummary.html
6. SweSum 2005 SweSum-Automatic text summarizer demonstrator.
   http://SweSum.nada.kth.se
7. Dalianis, H., M. Hassel. Development of a Swedish Corpus for Evaluating Summarizers and Other IR-Tools. Technical Report, TRITA-NA-P0112, IPLab-188, NADA, KTH, June 2001.
8. Dalianis, H., E. Åström. SweNam – A Swedish Named Entity Recognizer, Its Construction, Training and Evaluation. Technical Report, TRITA-NA-P0113, IPLab-189, NADA, KTH, 2001.
9. Mazdak, N. Farsi-Sum, a Persian Text Summarizer. Master Thesis of Nima Mazdak. Stockholm University, Sweden, 2004.
10. Hassel, M. Exploitation of Named Entities in Automatic Text Summarization for Swedish. – In: Proceedings of NODALIDA'03 – 14th Nordic Conference on Computational Linguistics, 30-31 May 2003, Uppsala, Sweden.
11. Lin, Cy. Training a Selection Function for Extraction. – In: the 8th International Conference on Information and Knowledge Management (CIKM' 99), Kansa City, Missouri, 2-6 November 1999.
12. Veerappan, R., P. J. Antony, S. Saravanan, K. P. Soman. A Rule Based Kannada Morphological Analyzer and Generator Using Finite State Transducer. – In: Proceedings of International Journal of Computer Applications (0975 – 8887), Vol. **27**, August 2011, No 10.
13. Jagadish, S. K., K. G. Srinivasa, R. B. Eswara. Information Retrieval by Text Summarization for an Indian Regional Language. – In: Proceedings of IEEE NLP-KE'2010, 21-23 August 2010, Beijing, China.

14. M a n i, I., M. M a y b u r y. Advances in Automatic Text Summarization. Cambridge, MA, MIT Press, 1999. ISBN: 0262133598.
15. M a y b u r y, T. M., M. I n d e r j e e t. Automatic Summarization Tutorial Notes for the American/European Conference on Computational Linguistics (ACL/EACL'01). Toulouse, France, 8 July 2001.
16. D a l i a n i s, H., M. H a s s e l, K. d e S m e d t, A. L i s e t h, T. C. L e c h, J. W e d e k i n d. Porting and Evaluation of Automatic Summarization. – In: H. Holmboe, Ed. Nordisk Sprogteknologi, Årbog for Nordisk Språkteknologisk Forskningsprogram 2000-2004 Museum Tusculanums Forlag, 2004.
17. D e S m e d t, K., A. L i s e t h, M. H a s s e l, H. D a l i a n i s. How Short is Good? An Evaluation of Automatic Summarization. – In: H. Holmboe, Ed. Nordisk Sprogteknologi, Årbog for Nordisk Språkteknologisk Forskningsprogram 2000-2004. Museum Tusculanums Forlag, 2004.
18. S e k i, Y. Sentence Extraction by tf/idf and Position Weighting from Newspaper Articles. – In: Proceedings of the Third NTCIR Workshop, 2003.
19. P o r t e r, M. F. An Algorithm for Suffix Stripping Program. – White Rose Consortium e-Prints Repository, Vol. **14**, 1980, No 3, 130-137.