An Extractive Approach of Text Summarization of Assamese using WordNet

Chandan Kalita
Department of CSE
Tezpur University
Napaam, Assam-784028
chandan_kalita@yahoo.co.in

Navanath Saharia
Department of CSE
Tezpur University
Napaam, Assam-784028
nava_tu@tezu.ernet.in

Utpal Sharma
Department of CSE
Tezpur University
Napaam, Assam-784028
utpal@tezu.ernet.in

Abstract

Automatic text summarization means finding out the summary of one or more document by a computer program. The output text or the summary should contain the most important points of the original text without changing its meaning. In this report, we present an extractive approach of Text summarization of Assamese, a free word order inflectional Indic language, using WordNet. From our experiment, we got approximately 78% accurate result.

1 Introduction

Automatic text summarization means finding out the summary of one or more document by a computer program. The output text or the summary should contain the most important points of the original text without changing its meaning. With huge amount of information available on the World Wide Web, there is a pressing need to have "Information Access" systems that would help users by providing the relevant information in a concise, pertinent format. Two main category of text summarization are (Das and Martins, 2007).

- Extractive
- Abstractive

Extractive approach is a procedure of identifying important sections of the text and reproducing them as they are. There are no modifications done in the input text pattern. In this approach, there are mainly two steps- *Extraction* and *Fusion*. In the extraction step, important sections are identified and extracted sections are combined coherently in the fusion phase. In abstractive approach considerable amount of linguistic analysis is performed for the task of summarization. In this approach, important sections of the input text are identified and produced in a new way. In the abstractive approach new sentence are generated without changing the topic meaning. In this paper, we present certain aspects of "Text Summarization" and implement one extractive approach for Assamese language that is the easternmost Indo-European language with around 30 million speakers.

In the next section, we describe prior works in single and multi-document text summarization. Section 3 and 4 describe preprocessing phases: similarity measures used in our approach for summarization of Assamese text and obtained results respectively. Section 5 concludes our paper.

2 Literature Survey

2.1 Single document summarization

In the 1990s, with the advent of machine learning technique used in NLP, a series of seminal publications appeared that employed statistical techniques to produce document extracts.

Naive-Bayes Method:

In this method the program is able to learn from existing data. A classification function determines for each sentence whether the sentence should be included in the summary or not using a Naive Bayes classifier. In this approach a score is given to each sentence and only the *n* top sentences are extracted.

Rich Features based Method:

Lin and Hovy, 1997 studied the importance of a single feature- "sentence position". Just weighing a sentence by its position in the text, which the authors term as the "position method", is based

on the idea that texts generally follow a predictable discourse structure, and the sentences of the main topic tend to occur in certain predefined locations (e.g. title, abstracts, etc). However, since the discourse structure significantly varies over domains, the position method is not a good choice. Naive-Bayes method and Rich Features based methods are some example of sentence extraction based summarization approach. There are some other approaches whose working principles are different but use extraction based. For example Hidden Markov Model, Neural Networks Third Party Features etc. These are basically machine-learning methods.

Deep Natural Language Analysis Methods

In this category, all approaches involve complex natural language analysis techniques. None of these approaches solves the problem using machine learning, but rather uses a set of heuristics to create document extracts.

2.2 Multi-Document Summarization

Extraction of a single summary from multiple documents has gained interest since mid 1990s, most applications being in the domain of news articles. Several Web based news clustering systems were inspired by research on multi-document summarization, for example Google News, Yahoo News etc. This departs from single-document summarization since the problem involves multiple sources of information that overlap and supplement each other, being contradictory at occasions. Therefore, the key tasks are not only identifying and coping with redundancy across documents, but also recognizing novelty and ensuring that the final summary is both coherent and complete.

Abstraction and Information Fusion:

SUMMONS (Radev and MCKeown, 1998) is the first historical example of a multi-document summarization system. It takes multiple documents about a single event of narrow domain from various sources and produces a brief summary containing information about the event. Rather than working with raw text, SUMMONS reads a database previously built by a template-based message understanding system. The architecture of SUMMONS consists of two major components: a content planner that selects the information to include in the summary through combination of the input templates, and a linguistic generator that selects the right words to express the information in grammatical and coherent text.

Graph Spreading Activation:

Mani and Bloedorn, 1997 describe an information extraction framework for summarization, a graph-based method to find similarities and dissimilarities in pairs of documents. Although no textual summary is generated, the summary content is represented via entities (concepts) and relations that are displayed respectively as nodes and edges of a graph. Rather than extracting sentences, they detect salient regions of the graph via a spreading activation technique. A document is represented as a graph as follows:
Each node represents the occurrence of a single word (i.e., one word together with its position in the text). Each node can have several kinds of links: adjacency links (ADJ) to adjacent words in the text, SAME links to other occurrences of the same word, and ALPHA links encoding semantic relationships captured through WordNet. Besides these, PHRASE links bind together sequences of adjacent nodes, which belong to the same phrase and NAME, and COREF link stands for co-referential name occurrences.

Centroid-Based Summarization

Generally this type of approaches do not use a language generation module(Zhang and Li, 2009). All documents are modeled as bags-of-words. The first stage consists of topic detection, whose goal is to group together news articles that describe the same event. To accomplish this task, an agglomerative clustering algorithm is used that operates over the TF-IDF vector representations of the documents. The second stage uses the centroids to identify sentences in each cluster that are central to the topic of the entire cluster. The system is easily scalable and domain-independent.

3 Our Approach

We have developed a text summarizing method for Assamese, based on the use of a WordNet and a stop word list. Since no prior Assamese WordNet exists, we have to build the required WordNet for our experiment. To populate the Assamese WordNet database (Hussain et. al, 2011) (as there is no publicly available WordNet database for Assamese) we uses the following sources of data -

- Online Dictionary
- Chandrakanta Abhidhan

We also create a stop word list of 168 Assamese words. Since no Assamese WordNet is available on the internet, the WordNet database was very small; therefore, we added to it all the words of our test document by ourselves.

3.1 Preprocessing

In the file, *Put every sentence in a new line*. It will help to retrieve each sentence easily and it can be solved by breaking each sentence on some special character. *All the words should be in their root form*. This is needed because the WordNet contains only the root words of a language. To solve this problem we need a stemmer. Designing a stemmer is difficult because it involves lots of morphological analysis. In our experiment, we manually performed this job. If a word is not available in the WordNet due to not being in root form, we can adopt the following idea for finding similarity between such two words. If the word $W_1=a_1a_2a_3\ldots a_n$ and $W_2= b_1b_2b_3\ldots\ldots b_m$ and not present in the WordNet then the similarity between W_1 and W_2 is

$$\frac{2*|\text{length of matching character sequence of } W_1 \text{ and } W_2|}{|\text{length of } W_1| + |\text{length of } W_2|}$$

For example

W_1= মৰম (Maram : *love*)

W_2= মৰমৰ (Maramar : *of love*)

Similarity = (2*3) / (3+4) = = 0.85

In (Zhang and Li, 2009) the author propose a sentence similarity computing method based on the three features of the sentences, the *word form feature*, the *word order feature* and the *semantic feature*, using weights to describe the contribution of each feature of the sentence. Since our work is on Assamese language and it is *free word order language* we do not need to consider the *word order feature*. In the next section of this report, we discuss the similarity measures that we used.

To calculate the semantic similarity we used the Assamese WordNet (Hussain et. al, 2011) . To find the similarity of two words we first arrange the WordNet entry of these particular words tree wise. The tree structure is created according to the relation between the words. After that, we count the number of edges *N* between both the words. If there is no relationship between the words then N become very large (size of the WordNet). In that case, similarity will be approximately zero. But in case of synonym words there are no any edges between the words. But in that case similarity should be one. Therefore after deriving N we will calculate the similarity as follows.

$WSS(w_1,w_2)=1/(N+1)$ i.e. semantic similarity will proportional to 1/N.

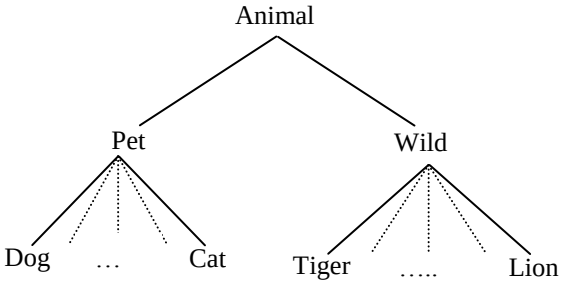

Figure 1: A fragment of WordNet

Similarity measure between sentences:

Definition 1: Word Form Similarity.

The word form similarity is mainly used to describe the form similarity between two sentences. It is the number of same words in two sentences measures it. First, we get rid of the stop words. If S_1 and S_2 are two sentences, the word form similarity is calculated by the formula (Zhang and Li, 2009).

$$Sim1(S_1,S_2)=2*(SameWord(S_1,S_2)/(Len(S_1)+Len(S_2))) \quad \ldots\ldots\ldots\ldots..(1)$$

Definition 2: Word Semantic Similarity.

The word semantic similarity is mainly used to describe the semantic similarity between two sentences. Here the word semantic similarity computing is based on the Assamese WordNet. Based on semantic similarity among words, (Zhang and Li, 2009) Word-Sentence Similarity (WSSim) is defined to be the maximum similarity between the word w and words within the sentence S. WSSim(w,S) is defined with the following formula

WSSim(w,S)=max{Sim(w, W_i)|W_i∈S, where w and W_i are words} ……..…..(2)

Here the Sim(w,W_i) is the word similarity between w and W_i. With WSSim(w,S), the sentence similarity is defined as follows:

$$Sim2(S_1,S_2) = \frac{\sum_{i=1}^{|S_1|} WSSim(w_i,S_2) + \sum_{j=1}^{|S_2|} WSSim(w_j,S_1)}{|S_1| + |S_2|}$$

……. (3)

In (3) S_1, S_2 are sentences; |S| is the number of words in the sentence S.

Definition 3: Sentence Similarity.

The sentence similarity is usually described as a number between zero and one, zero stands for non-similar, and one stands for totally similar. The larger the number is, the more the sentences similar. The sentence similarity between S_1 and S_2 is defined as follows (Zhang and Li, 2009):

$$Sim(S_1,S_2)=\lambda_1*Sim1(S_1,S_2)+\lambda_2*Sim2(S_1,S_2)$$

……. (4)

In (4) λ_1 and λ_2 are constants, and satisfy the equation $\lambda_1+\lambda_2=1$. λ_1 and λ_2 defines the contribution of the semantic similarity and word form similarity between S_1 and S_2. In our implementation we assumed $\lambda_1=\lambda_2=0.5$.

The function *Sim()* (Equation (4)) is used as the final measure of sentence similarity. Since we focus mainly on sentence clustering, we need a proper similarity function, which will measure the similarity between two sentences not only in terms of word level but also in semantic level. In this work during sentence clustering, we use this function as the similarity function. Next, we need to find the number of clusters.

3.2 Estimating the number of clusters

Determination of the optimal number of sentence clusters in a text document is a difficult issue and depends on the compression ratio of summary and chosen similarity measure, as well as on the document topics. For clustering of sentences, author used a strategy to determine the optimal number of clusters (the number of topics in a document) based on the distribution (Das and Martins, 2007) of words in the sentences:

$$k = n\frac{|D|}{\sum_{i=1}^{n}|S_i|} = n\frac{\left|\bigcup_{i=1}^{n} S_i\right|}{\sum_{i=1}^{n}|S_i|} \quad \text{……. (5)}$$

Where |D| is the number of terms in the document D, $|S_i|$ is the number of terms in the sentence S_i, n is the number of sentences in document D. Here terms refers to all those words which are not in stop word list. From the above formula we can see that if the number of terms which are common to some sentences are increased then the number of clusters will be reduced, i.e. common terms in multiple sentences means the domain of the document is small. Such documents contain less number of topic. Here we analyze the property of this estimation by two extreme cases.

Case 1:

The document is constituted of *n* sentences, which have the same set of terms. i.e. all the sentences are constituted of the same words. Therefore, the set of terms of the document coincides with the set of terms of each sentence i.e. D= (t1, t2, …, tm)=S_i=S. From (5) it follows that

$$k = n\frac{\left|\bigcup_{i=1}^{n} S_i\right|}{\sum_{i=1}^{n}|S_i|} = n\frac{\left|\bigcup_{i=1}^{n} S\right|}{\sum_{i=1}^{n}|S|} = n\frac{|S|}{\sum_{i=1}^{n}|S|} = 1$$

Case2:

The document consists of n sentence which do not have any term in common, that is, $S_i \cap S_j = \Phi$ for $i \neq j$. This means that each term belonging to *D* belongs only to one of the sentences S_i. i.e.

$$|D| = \left|\bigcup_{i=1}^{n} S_i\right| = \sum_{i=1}^{n}|S_i|$$

Therefore from (5) it follows that k=n. In both the extreme cases are depicted correctly. We assume that it will also work at any intermediate state. Therefore, we use the formula to find out the number of topics or the number of clusters of sentences in the document.

3.3 Summary Generation

After calculating the number of clusters, we use the K-means algorithm to cluster the sentences of the document. After clustering the sentences of the input document, the following few steps are there to find the final summary.

Step1: Extract the central sentences of each cluster.
Step2: Find similarity between headline (title) and those sentences, which are not included in the Step 1.
Step3: Add those sentences that are highly similar to the headline.
Step4: Sort them according to occurrence in the original input document.
Step5: Put the sorted sentences into the output document.

Based on the result of clustering, suppose the sentences clusters are $D = \{C_1, C_2, \ldots\ldots, C_k\}$. First, determine the central sentence μ_i of each cluster based on the accumulative similarity between the sentence S_i and other sentences, and then calculate the similarity between the sentence Si and the central sentence μ_i. Assume that the similarity of central sentence μ_i as 1, sort the sentences based on their similarity weights, and choose the high weight sentences as the topic sentences. After finding out the topic sentences, we add some more sentences to the summary based on the high similarity with the headline. In the summary generation process, there are mainly two steps. In the first stage, we select the cluster sentence, which is used as pruning of duplicate data. And in the second stage i.e. in step 2 and 3 we add those sentence which are representing the main topic of the document. Our entire method contains three phases as follows-

Phase1: Stem the words to obtain their root forms.
(Performed Manually in our experiment)
Phase2: Create a table of similarity between every sentence. (Auto)
Phase3: Cluster the sentences. (Auto)
Phase4: Summary Generation (Auto)

4 Experiments and Results

We conducted experiments to evaluate the performance of the automatic text summarization system based on sentences clustering. Automatic text summarization systems and evaluations of summary is not a straight-forward process. For evaluate the results we use F-measure which is widely used in Information Retrieval. Due to lack of large WordNet database, we restrict our experiment to a few small documents as input to find the summary. Table 1 gives the obtained result of 10 media documents with different domain.

Document	P	R	F
1	0.73	0.84	0.78
2	0.75	0.75	0.74
3	0.82	0.93	0.87
4	0.79	0.73	0.75
5	0.81	0.86	0.83
6	0.85	0.77	0.8
7	0.82	0.77	0.79
8	0.75	0.8	0.77
9	0.87	0.9	0.88
10	0.72	0.61	0.66

Table 1: Obtained Result

5 Conclusion

We have presented the approach to automatic text summarization based on the sentences clustering and extraction. The main contribution of this report is that it proposed and implemented a sentence similarity computing method based on the semantic features of the sentences, based on analyzing the word form Assamese WordNet (Hussain et. al, 2011). We find that the approach produces good result and can be considered for further improvement.

References

D. Das and A. Martins, "A Survey on Automatic Text Summarization". Literature Survey for the Language and Statistics" 2007

C.-Y. Lin, and E. Hovy, "Identifying topics by position", In Proceedings of the Fifth conference on Applied natural language processing, PP: 283-290, 1997

D. R. Radev, and K. McKeown, "Generating natural language summaries from multiple online sources". Computational Linguistics, 24(3), PP: 469-500, 1998

I. Mani, and E. Bloedorn, “Multi-document summarization by graph search and matching. In AAAI/IAAI, PP 622-628, 1997

P. Zhang and C. Li., Automatic Text Summarization Based on Sentences Clustering and Extraction. In Proceedings of IEEE International Conference on Computer Science and Information Technology, PP 167–170, 2009

I. Hussain, N. Saharia, U. Sharma "Development of Assamese WordNet" Machine Intelligence:Recent Advances, Narosa Publishing House, Editors. B. Nath, U. Sharma and D.K. Bhattacharyya, ISBN-978-81-8487-140-1, 2011

Online Multilingual Dictionary of North-East India, http://www.xobdo.net

Chandrakanta Abhidhana (Online), http://dsal.uchicago.edu / dictionaries/candrakanta/