Generic Text Summarization using WordNet

Kedar Bellare, Anish Das Sarma, Atish Das Sarma, Navneet Loiwal, Vaibhav Mehta, Ganesh Ramakrishnan, Pushpak Bhattacharyya

Department of Computer Science and Engg.
Indian Institute of Technology, Bombay
Powai, Mumbai, India
{kedarb,anish,atish,navneet,vaibhav,hare,pb}@cse.iitb.ac.in

Abstract

This paper presents a WordNet based approach to text summarization. The document to be summarized is used to extract a "relevant" sub-graph from the WordNet graph. Weights are assigned to each node of this sub-graph using a strategy similar to the Google Page-ranking algorithm. These weights capture the relevance of the respective synsets with respect to the whole document. A matrix in which each row repesents a sentence and each column a node of the sub-graph (i.e., a synset) is created. Principal Component Analysis is performed on this matrix to help extract the sentences for the summary. Our approach is generic unlike most previous approaches which address specific genres of documents like *news articles* and *biographies*. Testing our system on the standard DUC2002 extracts shows that our results are promising and comparable to existing summarizers.

1. Introduction

Text summarization finds varied applications in today's world. Some notable ones are: search engine hit summarization (summarizing the information in a hit list retrieved by some search engine); physicians' aids (to summarize and compare the recommended treatments for a patient); generating the blurb of a book ; and so on. Building automated summarizers can be very helpful in many such applications and saves a lot of manual work. An extract is a summary containing only sentences from the text and involves no natural language generation. Given a piece of text, our aim is to select the most "representative" sentences which will form the summary. A good summary should ideally have the following features: Relevance to the text, Informativeness and Conciseness.

1.1. Related work

In this section we cite relevant past literature, that use various summarization techniques.

The most common and recent text summarization techniques use either *statistical approaches*, for example (Strzalkowski, 1998), (Berger and Mittal, 2000), (Zechner, 1996), (Carbonell, 1998), (Nomoto, 2001); or *linguistic techniques*, for example (Klavans, 1995), (Radev, 1998), (Nakao, 2000), (Marcu, 1997); or some kind of a linear combination of these: (Goldstein et al., 1999), (Mani, 2002) and (Barzilay, 1997). Our algorithm is markedly different from each of these and tries to capture the semantics of the text.

There has also been a lot of work on text summarization using various kinds of *supervised learning* techniques starting with (Kupiec, 1995) and then further studied in (Teufel, 1997), (Mani, 1998), (Chuang, 2000) and (Berger, 2000); and (Amini and Gallinari, 2002) proposed a *semi-supervised learning* technique.

(Zha, 2002) developed an algorithm for summarization based on a mutual reinforcement principle where keyphrases and sentences are ranked based on their saliency scores.

(Goldstein et al., 1999) studies news article summarization and uses statistical and linguistic features to rank sentences in the document.

(Mani, 2002) has a study of various text summarization techniques. These include: location of a term in the document, presence of statistically salient terms, presence of cue phrases and connectivity of test units based on proximity, repetition, etc and taking a weighted average of these to obtain a summary.

We note that none of the above approaches to text summarization selects sentences based on the semantic content of the sentence and the relative importance of the content to the semantics of the text. Our algorithm is based on identifying semantic relations and is for generic text summarization unlike almost all previous ones.

1.2. A brief on the WordNet

WordNet (Fellbaum, 1998) is an online lexical reference system in which English nouns, verbs, adjectives and adverbs are organized into synonym sets or *synsets*, each representing one underlying lexical concept. Noun synsets are related to each other through *hypernymy* (generalization), *hyponymy* (specialization), *holonymy* (whole of) and *meronymy* (part of) relations. Of these, (*hypernymy*, *hyponymy*) and (*meronymy*,*holonymy*) are complementary pairs.

The verb and adjective synsets are very sparsely connected with each other. No relation is available between noun and verb synsets. However, 4500 adjective synsets are related to noun synsets with *pertainyms* (pertaining to) and *attra* (attributed with) relations.

1.3. Our approach

We use WordNet to understand the links between different parts of the document; Subsequently extract the portion of the WordNet graph which is most relevant and contains the main ideas present in the document. 2.. The idea of first getting a global view of the whole document, even before beginning to rank sentences is what differentiates our approach from the rest and also makes it generic.

2. Text Summarization Algorithm

The heart of our algorithm is the computation of global semantic information which captures the overall meaning of the text using WordNet.

The algorithm has the following five steps:

1. **Preprocessing of the text:**

   - Break the text into sentences.
   - Apply part of speech tagging to the words in the text. This is essential to pick the correct meaning of the word in WordNet. Hence if the word "pant" is used in the text as a verb, we will not associate it with a form of clothing.
   - Identify *collocations* in the text. A collocation is a group of commonly co-occuring words, for example, "miles per hour". Identifying collocations helps in capturing the meaning of the text better than that of the individual words (just like any idiom).
   - Remove *stop words* like "the", "him" and "had" which do not contribute to understanding the main ideas present in the text.

   The sequence of the above operations is important since we must identify collocations before removing stop words as many stop words often form part of collocations.

2. **Constructing sub-graph from WordNet:** We find the portion of the WordNet graph which is *relevant* to our text, i.e. those synsets whose meaning occurs in the text.

   First we mark all the words and collocations in the WordNet graph which are present in our text. We then traverse the generalisation edges of the WordNet graph starting from these marked words, and keep marking all the synsets which are reachable from the marked words. We do a breadth-first search and traverse the graph only till a fixed depth, as the meanings of synsets become too general to be considered thereafter. Finally, we construct a graph G containing only the marked words and marked synsets as nodes and the generalization links between them as the edges.

3. **Synset Ranking:** The basic motivation of this step is to rank the synsets based on their relevance to the text. So, if lots of words in the text correspond to the same synset, that synset or 'meaning' is more relevant to the text, and thus, it must get a higher rank. This idea has been borrowed from (Ramakrishnan and Bhattacharya, 2003), which details the use of WordNet Synsets as a mode of text representation.

   We construct a row vector R with an entry corresponding to each node of the graph. The i^{th} entry of R denotes the rank or "importance" of the i^{th} node in the graph, assuming the nodes are numbered from 0 to $n-1$, where n is the number of nodes. We also construct a square matrix A such that

   $$A[i,j] = \frac{1}{num\_predecessors(j)}, \text{ if } j \text{ is a child of } i$$
   $$0 \qquad otherwise$$

   where i and j are nodes in the graph G.

   Each entry in R is initialized to $\frac{1}{\sqrt{n}}$ Then, we keep applying the following transformation to R until the change in modulus of R becomes less than a fixed tolerance value.

   $$R_{new} = \frac{R_{old} * A}{|R_{old} * A|}$$

   After each iteration of the above transformation, we add a positive bias to the rank of the nodes corresponding to the words present in the text. This ensures that the rank of the words do not become too low.

4. **Sentence Selection:** The synset ranking algorithm gives us information about which synsets or meanings are more relavant to the text, the relevance being reflected in the rank of the synset.

   Now, we construct a matrix M with m rows and n columns, where m is the number of sentences and n is the number of nodes in G. Initially each M_{ij} is 0. Now, for each sentence S_i, we traverse the graph G starting with the words present in S_i and following the generalisation edges similar to step 2. We find out the set of reachable synsets SY_i. Now, for each synset $sy_{ij} \in SY_i$, we set $M[S_i][sy_{ij}]$ to the rank of sy_{ij} calculated in step 3.

   Now, for a sentence S_i and synset sy_j, we have

   $$M[S_i][sy_j] = 0 \quad \text{if } sy_j \text{ is not reachable from } S_i$$
   $$= R[sy_j] \quad \text{otherwise}$$

   Intuitively, each row of the matrix represents the meaning captured by the corresponding sentence out of the meaning of the whole text.

   Principal Component Analysis(PCA): We apply PCA on the matrix M and get a set of principal components or eigenvectors. The eigenvalue of each eigenvector is a measure of the relevance of that eigenvector to the meaning of the text. We sort the set of eigenvectors obtained according to their eigenvalues. Now, for each eigenvector, we find its projection on all the sentences and select the sentences with the highest projection. We select $n_{numselect}$ top sentences with high projection values in the summary, where $n_{numselect}$ is proportional to the eigenvalue of the eigenvector. If number of sentences to be selected is N, then $n_{numselect}$ corresponding to eigenvector i is $n_{numselect} = (\lambda_i / \sum_j (\lambda_j)) * N$ where λ_i is the eigenvalue corresponding to the ith eigenvector.

   The sentences we thus obtain are in order of their relevance to the text. A very notable feature of this method is that we avoid selecting too many sentences with similar semantic content. Moreover, sentences picked will not represent a similar meaning or semantic content.

5. **Final filtering**

   The last stage of our algorithm involves the application of simple heuristics to filter out the sentences which have undefined references. The heuristics applied are: removing sentences which contain words like "He", "It" etc. at the beginning and removing sentence which begin with quotes.

3. Semantic Interpretation of Computational steps

The first step of the algorithm is pre-processing of the text. The requirement of this has already been mentioned above.

The second step is synset ranking. Here, we start with the words present in the text and then draw edges moving upward to synsets they relate to. So, even if different synonyms of a particularly synset is present in the document, there will be many incoming edges and therefore that synset will get a higher weight. Also, this helps in word sense disambiguation. Take for example the word bank. Now if in the text there are other words like shore, sea etc. then these shall relate and connect to the bank synset relating to shore rather than that to financial institution.

Once the synset ranking has been done, sentence selection is done via PCA. As mentioned above, we have a matrix (call it A) with sentences as rows and synsets as columns. Semantically, this matrix therefore represents which sentences have which synsets or the other way round, which synsets are reflected in which sentences.

In effect, what has been done so far is to identify regions of influence of all the eigen-vectors. Then from each of these similarity islands, representative sentences are chosen. Each island is a set of sentences that closley relate to an eigen vector.

Then on performing AA^t we get a square matrix with as many rows/columns as sentences. This matrix gives us the "cosine similarity" between these sentences. Now, consider an n dimensional graph where n is the number of rows here. And then plot the columns as points on this graph. Now what does each point mean? A particular point shows the correlation of that sentence with various other sentences. The most prominent eigenvector of this square matrix, in terms of the graph will be that direction which has many points on the vector. This vector represents a hypothetical sentence, which is closest to all sentences in the document. We pick a sentence closest to this, because it is this sentence which best represents all the sentences in the document! The same is then done for the remaining eigenvectors also.

4. Illustration of the algorithm

We generated a summary for the following text document. This summary has been generated without the final filtering to illustrate the effectiveness of our approach. The document is:

Bombay is a busy city. Thousands of people come here every day in search of livelihood. No other city in India offers so much of employment opportunity. Since 1970 the population in the city has been steadily hovering around one crore. Next to Bombay come Calcutta, Delhi and Chennai. But, none of these have the dynamism and the charisma that is characteristic of Bombay. It is truly the international city of the country, truly cosmopolitan, truly multi ethnic. No city other than Singapore in South Asia surpasses Bombay in business volume. Probably Dubai comes close, but a very distant close. Indeed Bombay is a very active place.

The summary generated by our approach is quite good:
Bombay is a busy city. Thousands of people come here every day in search of livelihood. No other city in India offers so much of employment opportunity. Next to Bombay, come Calcutta, Delhi and Chennai. But none of these have the dynamism and charisma that is characteristic of Bombay. Indeed Bombay is a very active place .

	Copernicus	Ours
DUC-200	0.7864	0.7402
DUC-400	0.8402	0.7875

Table 1: Average Similarity Measure

5. Results

We tested our summarizer using the DUC'2002 multi-document summary coprus. To compare our summaries with those of the well known Copernicus summarizer (Cop,), we use DUC's human generated summaries as the benchmark. The results are very encouraging. DUC-200 and DUC-400 denote the DUC summaries of length 200 words and 400 words respectively.

Table 1 gives the average similarity between DUC-200 and DUC-400 with the summaries generated by our algorithm and that of Copernicus. The similairy measure used is the cosine similarity.

Figure 1 plots the similarity between our summaries with DUC-200 and that of Copernicus' with DUC-200. Figure 2 plots the similarity between our summaries with DUC-400 and that of Copernicus' with DUC-400. Dotted plots are that of Copernicus' while ours are in dashed.

In the figures the document number is plotted on the $x-axis$ and the cosine similarity on the $y-axis$. The horizontal lines in the plots show the average similarity.

6. Conclusion

There is an ever-increasing need for better automatic text summarization systems with the explosion in the amount of information available. We propose an algorithm for a generic text summarizer which selects sentences on the basis of their semantic content and its relevance to the main ideas contained in the text. Our algorithm compared well with the DUC'2002 data sets and their reference summaries. Even though our average results are slightly worse

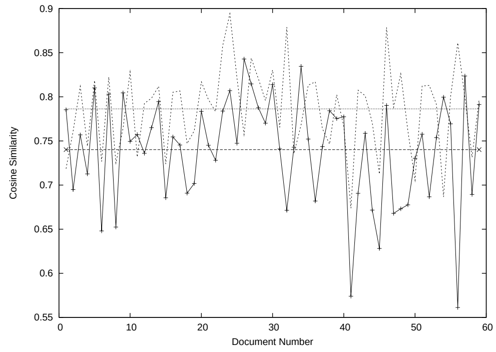

Figure 1: Similarity with DUC-200.

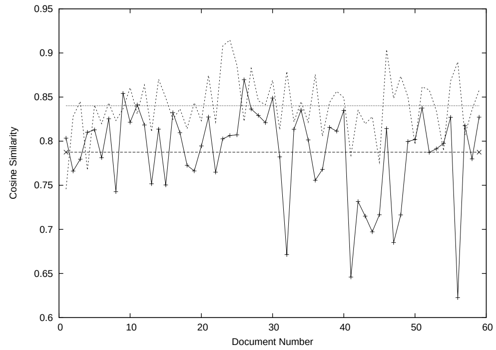

Figure 2: Similarity with DUC-400.

than that of Copernicus', our algorithm is simple, repeatable and the results can be verified, unlike Copernicus'. Moreover, ours is a novel approach and therefore, building on this could improve results drastically; further, this approach could be of independent interest.

7. Future Work

A number of interesting possibilities remain that we hope to explore in future. Firstly, the parameters used for generating summariers, eg, weightage given to different parts of speech, can be learned given a corpus of documents. Then, Resolution of pronouns can be used on top of the WordNet approach to get summaries which are more readable and have less dangling anaphors. Also, since the approach we have used is not language specific, we can use a lexical resource like WordNet for other languages and extend the summarizer to them.

8. References

Copernicus summarizer:. *http://www.copernic.com/en/products/summarizer/*.

Amini and Gallinari, 2002. The use of unlabeled data to improve supervised learning for text summarization. *Proceedings of the 25th ACM SIGIR*.

Barzilay, Elhadad, 1997. Using lexical chains for text summarization. *Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization.*.

Berger and Mittal, 2000. Query-relevant summarization using faqs. *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*.

Berger, Mittal, 2000. A system for summarizing web pages. *Research and Developement in Information Retrieval*.

Carbonell, Goldstein, 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. *Proceedings of the 21st ACM SIGIR*.

Chuang, Yang, 2000. Extracting sentence segments for text summarization: a machine learning approach. *Proceedings of the 23rd ACM SIGIR.*.

Fellbaum, Christiane (Ed.), 1998. *Wordnet : An Electronic Lexical Database (Language, Speech and Communication)*. MIT Press.

Goldstein, Kantrowitz, Mittal, and Carbonell, 1999. Summarizing text documents: Sentence selection and evaluation metrics. *Proceedings SIGIR*.

Klavans, Shaw, 1995. Lexical semantics in summarization. *Proceedings of the First Annuwal Workshop of the IFIP working Group for Natural Language Processing and Knowledge Representation*.

Kupiec, Chen, Pedersen, 1995. A trainable document summarizer. *Proceedings of the 18th ACM SIGIR.*.

Mani, 2002. Automatic summarization. *A tutorial presented at ICON*.

Mani, Bloedom, 1998. Machine learning of generic and user focussed summarization. *Proceedings of the Fifteenth National Conference on AI.*.

Marcu, 1997. From discourse structures to text summaries. *Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization.*.

Nakao, 2000. An algorithm for one-page summarization of a long text based on thematic hierarchy detection. *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*.

Nomoto, Matsumoto, 2001. A new approach to unsupervised text summarization. *Proceedings of the 24th ACM SIGIR*.

Radev, McKeown, 1998. Generating natural language summaries from multiple online sources. *Computational Linguistics*.

Ramakrishnan and Bhattacharya, 2003. Text representation with wordnet synsets. *Eight International Conference on Applications of Natural Language to Information Systems (NLDB2003)*.

Strzalkowski, Wise, Wang, 1998. A robust practical text summarization system. *Proceedings of the Fifteenth National Conference on AI*.

Teufel, Moens, 1997. Sentence extraction as a classification task. *Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization.*.

Zechner, 1996. Fast generation of abstracts from general domain text corpora by extracting relevant sentences. *COLING*.

Zha, 2002. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. *SIGIR*.