Automatic Summarizer to Aid a Q/A System

Rajarshi Das
7th Semester B.Tech
Department of Computer Science and Engineering
National Institute of Technology Calicut
Kozhikode 673601
Kerala, INDIA

Ajeesh Elikkottil
7th Semester B.Tech
Department of Computer Science and Engineering
National Institute of Technology Calicut
Kozhikode 673601
Kerala, INDIA

ABSTRACT
In this paper we couple a Q/A system with a summarizer and claim that an automatic summarizer coupled with a Q/A system will increase the efficiency of the Q/A system. Coupling a question-answering system with a text-summarization system may make the question-answering system more efficient without significantly altering the quality of results. This paper investigates methods to develop an extractive summarizer which aims to select the most important sentences which could be a possible answer to a given query.

Categories and Subject Descriptors
I.2.7 [ARTIFICIAL INTELLIGENCE]: Natural Language Processing – Text Analysis.

General Terms
Algorithms, Measurement, Performance, Design.

Keywords
Up-Keys, Summarization, Q/A System, Key-Words

1. INTRODUCTION
Auto-Summarization is indispensable with ever increasing volumes of valuable information. Spark-Jones [11] discussed several ways to classify summaries. The following three factors are considered to be important for text summarization.

Input factors : text length, genre, number of documents
Purpose factors : audience, purpose of summarization.
Output factors : running text or headed text etc.

The summarizer which we aim to make has the purpose of increasing the efficiency of a Q/A system. Summaries can be classified into different types based on dimensions, genres or context.

Dimensions : Single vs. Multi-document summarization
Genre : Headlines, outlines, minutes, chronologies, etc.
Context : Generic, Query specific summaries

Mani and Maybury[8] classifies summaries into extracts, where we select the most relevant sentences from the text, keeping in mind that the sentences could be answers for some possible questions on the document) and abstracts where text is analyzed, a conceptual representation is provided which in-turn is used to generate sentences for the summary.

An open domain Question Answering system (Q/A) aims at precisely answering a question asked by a user. The main challenge for a Q/A system is to answer precisely rather than providing a document containing the answer. We aim at simplifying this challenge by eliminating irrelevant information from the document and also by reducing the search-space. A lot of research work has been done on generic summarization [1 4 6] which aims at providing gist of the document to the readers. However, to the best of our knowledge, no notable work has been done which aims at improving the efficiency of a Q/A system by generating summaries of the document. In this paper we aim to develop an algorithm which will extract the sentences which contains possible answers to questions. First we aim to find the most important words (Up-Keys). We define Up-Keys as terms occurring in a document which reflect to a reader what the document is trying to convey. On the other hand Up-Keys may be possible answer to a question. For example : If we have a document on a cyclone which occurred at a particular place then the possible Up-Keys could be “cyclone”, the name of the cyclone, place where it occurred etc. If we look deeply then these words are already answers to questions such as “Where did the cyclone occur?” or “What was the name of the cyclone?” etc. Also lines containing Up-Keys may contain other valuable information which could also be possible answers to other questions. After finding Up-Keys we aim to select the most important sentences containing the Up-Keys. We developed a ranking algorithm which will select the most relevant sentences. Also it should be mentioned that this summarizer is different from a query-based summarizer. A query-based summarizer gives the answer as a summary to the reader who inputs the query. So summary can be made multiple times, but we will be summarizing the given document only once and try to give answers to as many queries as possible.
In section 2 the pre-processing techniques are presented. In section 3 the algorithms for finding the Up-Keys is described. In section 4 we describe the sentence selection algorithm. In section 5 the evaluation methods are described and in section 6 we conclude and mention some of our future work we plan to do.

In order to get the Synset type we had to get the Parts of Speech (POS) of the word. We got the POS for the word by using a statistical parser called the Charniak's parser [3]. A demo of the output given by the Charniak Parser for an input sentence is given below.

Input: Pratibha Patil is the president of India.
Output: (S1 (S (NP (NNP Pratibha)(NNP Patil))(VP (AUX is)(NP (NP (DT the) (NN president)) (PP (IN of) (NP (NNP India)))(. .))))

The tag set followed by the parser is the standard Penn Treebank Tag set. [9]. Since the WordNet is arranged in the form of Synsets so synonym replacement becomes quite an efficient task

2.2 Pronoun Replacement
An ordinary text document contains names, place names and object names and many other proper nouns which are later referenced using pronouns. These pronouns need to be resolved and replaced so that they contribute to the actual word count. For pronoun resolution we have implemented the Hobbs algorithm [7].

A demo of the Hobbs algorithm pronoun replacement is given below.

John and Mary lived in the big house. He built it with stone and clay. She also helped him.

Output:
John and Mary lived in the big house. John built the big house with stone and clay. Mary also helped John.

2.3 Stop Word Elimination
There are common words like {'a', 'an', 'the', 'is', 'are'} which are of no semantic relevance and carry no relevant information. So these stop-words are eliminated. A list of stop-words was collected from the Internet.

2.4 Stemming
After eliminating stop-words we applied a stemming algorithm (based on Porter's rules [10]). Stemming words allows to reduce the morphological variations of the word to their stem. So the words 'abound', 'abounded', 'abounding' will have the same stem 'abound'.

2.5 Sentence Separation
Given the text document we split the document into variable length sentences. Splitting into sentences is important because we aim to make an extractive summarizer where the summaries will be the collection of most important and relevant sentences from the document. We have used a tool for splitting sentences efficiently. The tool used is LingPipe sentence splitter. [2]

The pre-processing steps such as synonym resolution, pronoun resolution and stemming will increase the term-frequency of words since synonym will replace two similar meaning words by one, pronoun resolution will resolve a pronoun with the antecedent it is referring to and stemming will obviously reduce the morphological forms of a word into a stem.

3. FINDING UP-KEYS
After the pre-processing stage is over our next job is to identify the Up-Keys from the given input document (see Figure 1). The heuristics we follow aims at finding the most important terms from the document.

©2010 International Journal of Computer Applications (0975-8887)
Volume 1 – No. 19

Figure 1. A diagram showing the various components of our summarizer.
3.1 Term-Frequency
This is one of the important heuristic which we have applied. Here the claim is that terms which are important in a document will be occurring frequently. So counting how many times a term occurs in a document will help us finding the important terms in the document. It could be possible that certain terms in a document will be referred by its synonyms or by a pronoun (he/she/it etc). But we have already taken care of all these things in the pre-processing phase. So we can conclude that the term frequency obtained of each term will reflect the actual term-frequency of the term in the document.

3.2 Position
This heuristic claims that sentences which occur at the beginning of a document or the sentences which conclude the document are always important. This heuristic has been used in many summarization systems. [4] suggested that “improvements to the summaries can be achieved by weighting the sentences in the beginning of the articles more heavily.” They were mostly used to increase the rank of sentences. But here since our main aim is to find out important terms we add an extra score to each term occurring in the sentences.

After a lot of experimentation on standard documents we decided that if the length of the document is more than 100 sentences then we consider the first and last 3% of the document or we consider the first and last 5% of the document. All the terms which occur in this region are awarded a normalized score of unity.

3.3 Variance
Terms which occur throughout the document are of course more relevant than terms which are concentrated in few locations. It could also be possible that the terms may be the theme of the document (in which case it should be a part of the Up-Keys). But in most of the documents, instead of repeating the terms the terms could also be referred by its pronouns or synonyms. Yet again it proves that the pre-processing steps are indeed important. This heuristic basically tries to find out whether a term is sufficiently spread throughout the document. To calculate the variance weight we have considered the first sentence number and the last sentence number of the term in which it occurs. The variance weight will be directly proportional to the difference between the first and the last sentence number in which the term occurs. But if we only consider the difference we might get wrong inferences since in the worst case a term may occur in the first sentence and the last sentence of the document and it occurs nowhere else. Hence term-frequency of the term in the document is also significant in calculating the variance weight of a term. Hence we have used the following formula for calculating the variance weight of a term.

\[ Vwt(t) = (1 - f) \times TFwt(t) \]

where

\( f \) is the last sentence number in which the term ‘t’ occurs.
\( f' \) is the first sentence number in which the term ‘t’ occurs.

TFwt(t) is the term frequency of the term t in the given document.

3.4 Named Entity
A named-entity recognizer will classify each term in a document to a specific named entity such as PERSON or PLACE or DATE or MONEY etc. If the information that a term belongs to a named-entity is available to us then it would be of great importance. This is because many questions have answers which are named entities. For example many questions such as Who or When or Where have answers which are PERSONS, TIME and PLACE respectively. For named entity recognition we have used the LingPipe named entity recognizer [2]. Below is a demo of how LingPipe works.

Input: Bill Gates is the owner of Microsoft. He is from Washington.

Output:

<output><s i=0><ENAMEX TYPE="PERSON">Bill Gates</ENAMEX></s> is the owner of <ENAMEX TYPE="ORGANIZATION" >Microsoft</ENAMEX></s> <s i=1> He is from <ENAMEX TYPE="PLACE" >Washington</ENAMEX></s></output>

Many systems which have used this heuristic, often give different weight to named-entities depending on the situation. For example in some cases named-entity PERSON is given more score than other named-entity. But for our summarization system any sentence which could be possible answers to questions are considered to be important. So we have assigned a equal score of unity to each term which belongs to a named-entity.

The score of each term calculated from each heuristic is normalized. The total score of each term is evaluated by multiplying each heuristic with certain weights. Hence,

\[ \text{Score (t)} = w1*TFwt (t) + w2*Cwt (t) + w3*Vwt(t) + w4*Ewt (t) \]

where

TFwt(t) is the calculated term frequency of each term.
Cwt (t) is the position weight of each term.
Vwt(t) is the variance weight calculated for each term.
Ewt (t) is the named-entity weight given to each term.

While deciding upon the weights which had to be assigned to each heuristic we had to decide which heuristic is more important than the other. Clearly, the term-frequency factor would be the most important among all the above-mentioned heuristics. Also since our aim is to make a summarization system for a Q/A system therefore importance must also be given to the terms which are also named-entities. Since from the formula it is clear that the variance of a term is directly proportional to the term-frequency and since we already have decided that we will be giving the maximum weight to the term-frequency of a term therefore we decided to give the least weight to variance weight. Finally after a lot of experimentations with input documents we concluded on the weights as

Table 1. Weights decided for each heuristic.

<table>
<thead>
<tr>
<th>Term-Frequency</th>
<th>Position</th>
<th>Variance</th>
<th>Named-Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>8</td>
<td>4</td>
<td>10</td>
</tr>
</tbody>
</table>
Now from the above mentioned formula we can calculate the score of each term. The next important step was to select the Up-Keys which is the most important part of the whole project since better Up-Keys means a better hit-rate for a Q/A system.

Since score from each factors were normalized therefore the maximum score of any up-key would be 37 (15+8+4+10). To select the up-keys we sort the up-keys according to their scores in the decreasing order. At first we decided that we will select the first 6-7 terms as the Up-Keys but as we tested on a number of documents we found out that always selecting the first 6-7 Up-keys does not necessarily give good Up-keys since there can be a great variation among the weights of the terms selected. So we later decided that we will restrict the range of weights from [max_wt, max_wt-10] where max_wt is the maximum weight of any term in a document.

Another problem we faced was that within the range [max_wt, max_wt-10] there were more than the required number of Up-Keys but we cannot select all of them since there would be a problem when we try to select the sentences containing the Up-keys because in most cases it will span the entire document. So therefore the next challenge was to select the most important terms as the Up-Keys and to eliminate other terms. The algorithm we decided was

1. Select those terms which are named-entities and eliminate those which are not.
2. If the number of terms remaining is more than the required number then go to step 3 else we have found our Up-Keys
3. Select those terms which have higher positional score (higher Cwt)
4. If the number of terms remaining is more than the required number then go to step 5 else we have found our Up-Keys
5. Select those terms which have higher variance weight.
6. Select the remaining terms as the Up-Keys.

The remaining terms which were present were considered as Up-Keys. Our first elimination criterion was based on the fact whether the terms were named-entities or not. The main reason behind this was as mentioned earlier terms which were named-entities could be probable answers to some questions. If the number of terms left were still more than the required number then our next elimination criterion was whether the terms occurred at important positions in the document. Lastly if there were still more number of terms then we eliminated with respect to higher variance weights. The reason for selecting variance weight over frequency was because we had already given the highest weight to frequency and least weight to variance while calculating the scores of terms.

4. SENTENCE SELECTION

After selecting the Up-Keys our next job is to select the relevant sentences from the document (see Figure 1). We select the most important sentences containing the Up-Keys. The default summary length we choose is 25-30% of the total length of the document. The algorithm we follow is:

Let the length of the document be L sentences. Our algorithm aims to select the most relevant 0.25L – 0.3L sentences. We call this length the optimal length. We calculate for each sentence the number of Up-Keys present and we group them in order of the number of Up-Keys present in them. Thus we form several groups of sentences containing equal number of Up-Keys.

At this stage the cosine similarity of the sentences with each other within each group were measured in order to remove redundancy. The redundancy is removed at this stage because sentences containing redundancies will most probably in the same group. Also checking redundancies in the same group will make the system more efficient since in most cases we will be comparing a few numbers of sentences. Once the cosine similarity measures between two sentences were found to be above 0.8, the sentence with the lesser length was eliminated.

The formula for cosine similarity is $s^2 / (A*B)$

Where, ‘s’ is the number of similar words and ‘A’ and ‘B’ are the number of words in the two sentences.

The sentence selection algorithm is then applied to the redundancy removed groups of sentences. The algorithm is:

Let $G_1, G_2, G_3, \ldots, G_n$ be the groups of sentences formed where each sentences in group $G_i$ contains the same number of Up-Keys. Let $S = \{k_i | k_i$ is the number of Up-Keys in set $G_i\}$. Let max(S) denotes the maximum element present in S.

1. Let m = max(S). Select the group of sentences which contain m number of Up-Keys in them. $S = S - \{m\}$.
2. If the number of sentences selected is within the optimal length then we have selected the required sentences for the summary. If the no of sentences selected is less than the optimal length then repeat step 1. It is also possible that the number of sentences selected becomes more than the optimal length. In that case we go to step 3.
3. Since we have more number of sentences than required therefore we will have to eliminate certain number of sentences. Since all the sentences in the last group selected has the same number of Up-Keys we cannot differentiate on the basis of Up-Keys (We give similar importance to all the Up-Keys selected). The algorithm which we apply ranks the sentences by adding the weight of the terms in those sentences which are not Up-Keys. We took the sum of the scores of the four heuristics mentioned above (see Section 3). We rank the sentences in the decreasing order of the scores and select the required number of sentences with the higher ranks.

After selecting the required number of sentences we concatenate them in the order in which they appear in the original document. As with other summarization system coherence and other factors are not of prime importance since we are trying to improve the hit rate of a Q/A system.

5. EVALUATION

One of the standard measures of evaluating summaries is by calculating the recall and precision scores. To understand recall and precision let us define some terms,

Let correct = the number of sentences extracted by the summarizer and are also present in the manual summary of the document.

wrong = the number of sentences extracted by the summarizer but is not present in the manual summary of the document.

missed = the number of sentences present in the manual summary but not in the summary generated by the summarizer.

Precision = correct/ (correct + wrong)
Recall = correct/ (correct + missed)
F-score = 2*Precision*Recall/ (Precision + Recall)
30 standard Brown documents were selected from [12]. These documents were manually summarized by a group of 5 independent people. They were asked to make summaries of 25% of sentences. These summaries were compared with the summaries generated by the summarizer. The average recall score obtained was 0.37 and the precision score was 0.39.

Hence the F-score is 0.3797. Though this is not the best score obtained by any summarizer but as we mentioned earlier this is a summarizer to aid an Q/A system so precision and recall method is not the best way to evaluate it. The most appropriate way of evaluating would be by using any existing Q/A system but unfortunately we did not have a proper Q/A system to experiment.

To measure the relevance of the Up-Keys selected we compared our summarizing system with an existing commercial summarizing system Copernic Summarizer, which also gives the set of key-words it selected for making the summaries. We compared the Up-Keys with the key-words of the Copernic summarizer.

We calculated the F-score by

Let \( N_c \) be the number of common key words selected by both the summarizers.
\( N_t \) be the total number of key-words selected by our summarizer
\( N_s \) be the total number of keywords selected by Copernic Summarizer.

Precision = \( \frac{N_c}{N_t} \)
Recall = \( \frac{N_c}{N_s} \)
F-Score = \( \frac{2 * N_c}{N_t + N_s} \)

The same 30 Brown documents were used for evaluation. The average precision obtained was 0.88 and the average recall score was 0.76. Hence the F-Score was 0.8156. This is a promising result.

6. FUTURE WORK

If the system could be made intelligent enough to find the questions which it could answer from the summary it generated then any match of the questions (whose answers are present in the summary) with the query could be answered by the Q/A system very efficiently. Furthermore our summarizer would perform poorly if the query asks for a descriptive answer.

7. ACKNOWLEDGMENTS

Our sincere thanks to Dr Ajai Jain, Professor, Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur for guiding us in our project.

8. REFERENCES