Generating Text Summaries through the Relative Importance of Topics

Joel Larocca Neto, Alexandre D. Santos, Celso A.A. Kaestner and Alex A. Freitas

PUC-PR, PPGIA, CCET , Rua Imaculada Conceição, 1155,
80215-901 Curitiba, PR, Brazil
{joel, kaestner, denes, alex}@ppgia.pucpr.br

Abstract. This work proposes a new extractive text-summarization algorithm based on the importance of the topics contained in a document. The basic ideas of the proposed algorithm are as follows. At first the document is partitioned by using the TextTiling algorithm, which identifies topics (coherent segments of text) based on the TF-IDF metric. Then for each topic the algorithm computes a measure of its relative relevance in the document. This measure is computed by using the notion of TF-ISF (Term Frequency - Inverse *Sentence* Frequency), which is our adaptation of the well-known TF-IDF (Term Frequency - Inverse *Document* Frequency) measure in information retrieval. Finally, the summary is generated by selecting from each topic a number of sentences proportional to the importance of that topic.

1 Introduction

The summarization task consists of condensing the contents of a document, preserving its essential ideas. With the fast growth in the amount of textual information available on-line, there is clearly a strong need for automatic summarization systems. One of the advantages of this approach is that, given a summary, the user can judge whether or not the full document should be analyzed, so saving a precious time for the user.

Even in the case of texts which already contain summaries written by the text's author, the automatic generation of summaries has some unique advantages, such as: (a) It is possible to generate a summary with the size specified by the user, with the desired level of granularity - unlike manually-written summaries, which are static; (b) It is possible to create links between a sentence of the summary and a corresponding block of sentences in the full text.

In essence, the summarization task can be divided into two broad phases: (a) Construction of a representation of the text; (b) Generation of the summary, which can include extraction of sentences and/or construction of new sentences. Since the automatic construction of new sentences is an extremely difficult task, most systems generate a summary by extracting the most relevant sentences from the original document. This approach, which is the one followed in this paper, is called extractive summarization.

This paper proposes a new extractive text-summarization algorithm that first partitions the document into a list of topics and then selects the most relevant sentences from each topic. The proposed algorithm is quite flexible, due to two factors. First, the number of sentences selected from each topic is proportional to the relative importance of the topic within the document. Second, the algorithm can generate a summary with the size desired by the user.

This paper is organized as follows. Section 2 presents an overview of the system. Section 3 describes the structure of a typical input document for our system. Section 4 describes the basic ideas of the TextTiling algorithm and the adaptations that we have performed in it. Section 5 introduces our method to compute the importance of each topic identified by our modified version of TextTiling. Section 6 introduces our method for selecting the most relevant sentences from each topic. Section 7 discusses computational results. Finally, section 8 concludes the paper.

2 Overview of the System

At first our system partitions the document into topics, and then it assigns an importance weight to each topic. Given the user-specified number of sentences to be selected for the summary, the system selects the most relevant sentences from the most important topics. As shown in Figure 1, the system processing can be divided into four phases, as follows:

***Preprocessing* -** The document is first converted to a well-known bag-of-words representation [3]. In essence, for each word the system stores the observed frequency of that word in the document. However, not all words occurring in the document are included in the bag-of-words representation. For texts in English our system performs the well-known preprocessing steps of case folding, stopword removal and stemming [9]. Stopword removal is performed by using a list of 524 words obtained from the source code of the BOW library, developed by Carnegie Mellon University. Stemming is performed by Porter's algorithm [7]. For texts in languages other than English our system uses the n-gram representation [1], more precisely quad-grams. This representation is language-independent, so that it avoids the need for using language-dependent lists of stopwords and stemming algorithms.

Partitioning the Document into Topics - In order to partition the document into topics we used a modified version of the TextTiling algorithm, described in section 4.

***Computation of Topic Strengths (Relative Importance of Topics)* -** For each topic identified in the previous phase our system computes a topic importance value, i.e. a measure of the relative importance (in terms of percentage) of the topic within the document. This importance value is computed by using the notion of TF-ISF (Term Frequency - Inverse *Sentence* Frequency), which is our adaptation of the TF-IDF (Term Frequency - Inverse *Document* Frequency) measure, as described in section 5.

***Selection of the Most Relevant Sentences from the Topics* -** The selection of the most relevant sentences from the topics is based on the topic strength computed for each topic (the stronger the topic is, the more sentences it will provide for the summary) and on the similarity between each sentence and the centroid of its corresponding topic, as explained in section 6.

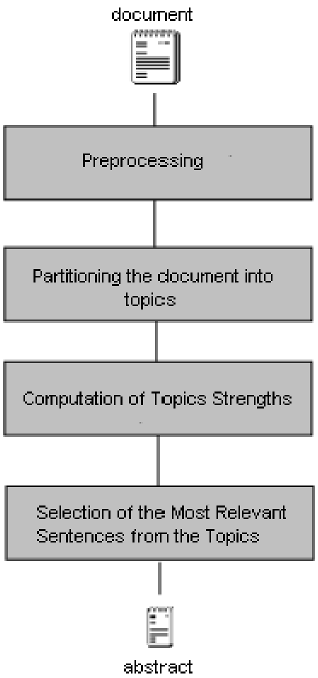

Fig. 1. Overview of the System

3 Structure of an Input Document

A typical input document for our system has the following structure:

```
<S>However, members from the Northwest Territories and Yukon
Territory voted against Meech Lake because it will make it
more difficult for their remote regions to become full
provinces.</S>
<S>Under the accord, the approval of all provinces will be
needed before the territories can obtain provincehood.</S>
<S>At present, it only requires approval by two-thirds of the
provinces containing at least 50 percent of the Canadian
population.</S>
```

Note that in the above example of input document the sentences have been previously separated, which is indicated by the tags <S> and </S> (denoting the beginning and the end of a sentence, respectively). The rationale for this option is twofold: (a) The use of simple methods for separating sentences, such as the detection of ".", "?", "!", is not very effective, since the determination of sentence boundaries is far from trivial [6]; (b) Large-scale projects for evaluating summarization algorithms, such as the TIPSTER Text Summarization Evaluation Conference (SUMMAC) [5], already use this kind of representation - which allows different summarization systems to be evaluated on a document with standard sentence boundaries.

It should be noted that not all documents contain a consistent topic structure. Some texts discuss a single subject. This is often the case in short texts extracted from newspapers, containing, say, from 10 to 20 sentences. The proposed algorithm gives better results in longer texts, containing, say, 30 or more sentences and having a well-defined topic structure, which is, for instance, the case in technical articles.

4 The TextTiling Algorithm

The TextTiling algorithm [2] proposes a method for partitioning documents into coherent units. This is performed by using a variation of the TF-IDF measure, a commonplace measure in information retrieval, as the basis for determining the similarity between two blocks of sentences, where each block usually contains from 3 to 5 sentences. This variation consists of replacing the notion of “document” with the notion of “block”. Hearst did not introduce any new terminology to denote this variant, referring to it simply as TF-IDF (Term Frequency – Inverse *Document* Frequency). However, we feel the use of the term is somewhat misleading, since “blocks” and “documents” are two different kinds of textual entities. Thus, to avoid confusion, in this paper, when we refer to the TextTiling algorithm we will use the term TF-IBF (Term Frequency – Inverse *Block* Frequency). In TextTiling the TF-IBF(w,b) value of a word w in a sentence block b is essentially the frequency of w in b divided by its frequency throughout the document (i.e., in all sentence blocks). The algorithm works in two steps.

First, all the pairs of adjacent blocks of sentences are compared and assigned a similarity value. The similarity between two sentence blocks is computed by the well-known measure of the cosine between two vectors. In TextTiling each vector coordinate corresponds to a TF-IBF(w,b) value. Hence, a high similarity value between two blocks indicates not only that the blocks have words in common but also that their common words are relatively rare with respect to the rest of the document.

Second, the similarity values are plotted against sentence numbers and smoothed in such a way that sudden quick dips are removed while at the same time the general trends of the graph are preserved - see [2] for details. The graph is then analyzed concerning the location of peaks and valleys.

Peaks indicate that the corresponding adjacent blocks cohere well, whereas valleys indicate a potential break of coherence, which suggests a frontier between two distinct topics. Actual values of similarity are not taken into account, the algorithm considers only the relative differences. The only adjustable parameter of the algorithm is the size (number of sentences) of the block used for comparison. This is heuristically set to the average length of the paragraphs in the document.

In order to compare the topics found by the algorithm with the topics found by human judges, the N-1 lowest valleys are selected by the algorithm as topic frontiers, where N is the number of topics identified by the human judges.

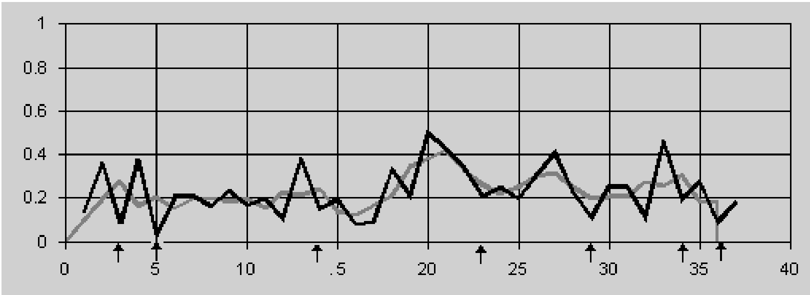

Fig. 2. Results of Text Tiling Algorithm in a document

Figure 2 shows the result of applying TextTiling to a document in Portuguese containing 110 sentences, using a block size of 3 sentences and using quad-grams (rather than stemming and stopword removal) as preprocessing. The black line represents the values of TF-IDF similarity between adjacent blocks, before smoothing. The gray line represents the same values after smoothing the black line. The arrows represent the valleys found by the algorithm *directly using the values of TF-IBF similarity (without smoothing)*, which led to better results in our experiments.

4.1 Extensions to the TextTiling Algorithm

In order to use TextTiling in our system we have extended it as follows. The algorithm must be able to decide how many topics occur in the document without using the result of any human judgement to provide the number of topics. In other words, we need to increase the autonomy of the algorithm.

In order to achieve this goal we have used an empirically-determined threshold to represent the break of similarity between blocks. Typical values of this threshold are between 0.1 and 0.2. With this extension, the algorithm for finding values in the graph of block similarities is as shown below, where sim(p) denotes the similarity between the block in the p-th position and the block in the (p+1)-th position in the text.

```
1) Scan the graph, beginning from position p := 0; direction := none; counter := 0;
2) IF sim(p+1) - sim(p) > 0
      THEN the line of similarity between blocks is rising;
           IF direction=falling and counter > threshold
              THEN add current block to the list of topics;
                   direction := rising;
3)    ELSE the line of similarity between blocks is falling;
           IF direction=rising
              THEN counter := sim(p+1) - sim(p);
                   direction := falling;
```

This simple algorithm allows the detection of topics whose relative fall of similarity is greater than the value of the pre-specified threshold.

In addition, we have modified the preprocessing of TextTiling. We have used the n-gram representation rather than stopword removal and stemming. As discussed in section 2, this renders the algorithm language-independent.

5 Determining Topic Importance Values

After partitioning the document into a list of topics the system must be able to compute the relative importance of each topic in the document, in order to extract more sentences from major topics than from minor topics. This is crucial to avoid the generation of irrelevant summaries. For instance, suppose that we wish to summarize a scientific paper. Topics such as "References" clearly should not be contained in the summary, and so should be assigned a low value of importance.

Our solution for determining the importance of each topic consists of two steps. First the algorithm calculates, for each topic, the summation of the importance of all sentences included in that topic. This requires a way to compute the importance of each sentence. Our solution for this problem is based on the idea of computing the average value of the TF-ISF (Term Frequency - Inverse *Sentence* Frequency) measure over all words of the sentence [4]. This measure is defined by replacing the notion of *document* in the TF-IDF (Term Frequency - Inverse *Document* Frequency) measure by the notion of *sentence*.

More precisely, the importance of each topic t, denoted TopImp, is computed by the formula below.

$$\text{TopImp(t)}= \sum_{s=1}^{|S(t)|}\left(\frac{\sum_{w=1}^{|W(s)|} TS-ISF(w,s)}{|W(s)|}\right), \qquad (1)$$

where w denotes a word, s denotes a sentence, |W(s)| is the number of words in sentence s, and |S(t)| is the number of sentences in topic t.

Once the importance of each topic is computed by the above formula, the relative importance of each topic (denoted RelTopImp) is obtained by simply normalizing the result of the above formula to get a number between 0 and 1, that is:

$$\text{RelTopImp(t)} = \frac{Top\,\mathrm{Im}\,p(t)}{\sum_{i=1}^{|T|} Top\,\mathrm{Im}\,p(i)} \qquad (2)$$

where |T| is the number of topics.

6 Selecting Relevant Sentences from Topics

In the previous steps the system has partitioned the document into topics and computed a measure of relative importance for each topic. As an example, a document of 24 sentences might be partitioned into 6 topics, as follows:

Table 1. Document Topics

Topic	Relative Importance	Sentences

1	0.24	1,2,3,4,5
2	0.36	6,7,8,9,10,11
3	0.10	12,13,14
4	0.05	15,16,17,18
5	0.20	19,20,21
6	0.05	22,23,24

The system must be capable of generating summaries of any size specified by the user. Hence, for a given summary size, the system must decide: (a) how many sentences will be selected from each topic; (b) for each of the topics, which sentence(s) will be selected from it.

The basic principle for answering the first question is, as mentioned before, that the number of sentences selected from each topic is proportional to that topic's relative importance. More precisely, to compute the number of sentences selected from each topic the algorithm performs two steps. First, it multiplies the topic's relative importance by the number of sentences to be selected for the summary. In the above example, assuming the user wants a summary with five sentences, for the topics numbered from 1 through 6 we would get the quantities 1.2, 1.8, 0.5, 0.25, 1.0 and 0.25, respectively. In the second step the algorithm rounds up/down these quantities for an integer number. This step is implemented by the following procedure:

```
/* t = number of topics */
/* N = user-specified number of sentences to be selected for the summary */
/* tot_sel = number of sentences to be selected for the summary */
/* sel_i = (not necessarily integer) number of sentences from topic i to be selected
        for the summary, computed by RelTopImp(i) * tot_sel;   */
/* int_sel_i = integer number of sentences from topic i to be selected for summary */
input: sel_i, i = 1,...,t
output: int_sel_i, i = 1,...,t
tot_sel = 0;
FOR i = 1,...,t
   int_sel_i = truncate(sel_i);   leftover_i = sel_i - int_sel_i;   tot_sel = tot_sel + int_sel_i;
ENDFOR
WHILE (tot_sel < N)
   find topic t such that leftover_t is the maximum over all leftover_i, i = 1,...,t
   int_sel_i = int_sel_i + 1;   leftover_i = 0;   tot_sel = tot_sel + 1;
ENDWHILE
```

Continuing the above example, the input for the above procedure would be:

sel_1=1.2, sel_2=1.8, sel_3=0.5, sel_4=0.25, sel_5=1.0 and sel_6=0.25.

After the execution of the FOR statement and before the first iteration of the WHILE statement we would have:

int_sel_1=1, int_sel_2=1, int_sel_3=0, int_sel_4=0, int_sel_5=1, int_sel_6=0.

After the execution of the first iteration of the WHILE statement we would have int_sel_2 = 2; and after the execution of the second iteration of the WHILE we would have int_sel_3 = 1. At this point tot_sel = N = 5, and so the algorithm ends outputing:

int_sel_1=1, int_sel_2=2, int_sel_3=1, int_sel_4=0, int_sel_5=1, int_sel_6=0.

Note that the above procedure just computes the number of sentences to be selected from each topic, but it does not answer the question of which sentence(s) is(are) selected from each topic.

The answer for this question is that, for each topic i, i =1,...,t, the system selects the int_sel_i sentences that are most similar to the centroid of that topic. The centroid of the topic represents the average TS-ISF vector of all sentences contained in the topic.

More precisely, each coordinate TS-ISF(i) of the centroid vector, i=1, ..., |w| - where |w| is the number of distinct words in the document, is given by the formula:

$$TS-ISF(i)=\frac{\left(\sum_{j=1}^{|S|} TS-ISF(i,j)\right)}{|S|}, \quad (3)$$

where TF-ISF(i,j) is the TS-ISF value of word i in sentence j, and |S| is the number of sentences in the topic.

The similarity between a sentence and the centroid of a topic is measured by computing the cosine of the angle formed by the two vectors (i. e. the sentence vector and the centroid vector), as usual in information retrieval [9].

7 Computational Results and Discussion

In this section we describe the results of an experiment carried out to evaluate our text summarization algorithm. In order to make our experiments more relevant, we have used the evaluation framework of a large scale project for evaluating text summarization algorithms: the TIPSTER Text Summarization Evaluation Conference (SUMMAC) – hereafter referred to as the SUMMAC project for short [5]. This is an important point, since there is no standard measure of summary quality. To quote [5, p. 3]: “Text summarization is still an emerging field, and serious questions remain concerning the appropriate methods and types of evaluation.”

Summary quality measures can be roughly categorized into intrinsic and extrinsic measures. The former evaluate the quality of a summary directly based on analysis of the summary, whereas the latter evaluate the quality of a summary based on how it affects the completion of some other task (e.g. determining the relevance of a document to a topic, in text categorization). Another categorization can be made between objective or subjective measures of summary quality.

To evaluate our text summarization algorithm, we have compared the summaries produced by our system against the summaries produced by algorithms performing the *adhoc* task of the SUMMAC project. This task is defined as follows [5, p. 4]:

“The real-world activity represented by this task is that of an analyst conducting full-text searches using an IR [information retrieval] system who must determine quickly and accurately the relevance of a retrieved document. Here the topic is provided as an input to the summarization system, and the evaluation seeks to determine whether the ... summary is effective in determining the relevance of the full-text source to a topic. Given a document (which could be a summary or a full-text source – the subject isn’t told which), and a topic description, the human subject determines whether the document seen is relevant to the topic. The accuracy of the subject’s relevance assessment decision is measured in terms of separately obtained ‘ground-truth’ judgments of the relevance of the full-text source to the topic, which were separately obtained from the Text Retrieval (TREC) ... conferences...”

It should be noted that this definition implies that, in this task, summary evaluation is both extrinsic and subjective. Ideally, we should evaluate the summaries produced by our algorithm using the same performance metrics as the above task of the SUMMAC project. In this case our algorithm could be directly compared against the 16 summarization algorithms that participated in the *adhoc* task of that project. In practice, unfortunately, this is unfeasible, since the conference is over, and we do not have access to all the infra-structure of the conference (including the human analysts who assessed the relevance of all the summaries produced by each of the algorithms competing in the conference).

Therefore, we have opted for carrying out an intrinsic subjective comparison of the summaries produced by our system against the summaries produced by the CGI/CMU and Cornell systems. These systems were chosen mainly because, overall, they produced the best and second best (respectively) results of the SUMMAC evaluation in the above-described *adhoc* task.

We have compared the summaries produced by the three systems – our, the CGI/CMU and Cornell systems - for several source texts.

The last two systems produced summaries limited to 10% of the number of characters in the text (spaces excluded). Our system, on the other hand, produces summaries limited to an arbitrarily-specified percentage of the number of lines in the text. We have tried to minimize the distortion caused by these diferents approaches by fixing the limit of the summaries in 10% of the number of lines in the text.

We have done experiments using papers randomly selected from the SUMMAC paper base. For each paper we produced a summary using our system and obtained the summaries produced by the CMU and Cornell systems for the adhoc task of the SUMMAC project. Then the relative quality of the three summaries was evaluated by a human judge. (Although this judge is not a native English speaker, she is a graduate in Linguistics and has several years of experience as a full-time English teacher.) For each paper, the human judge evaluated the three summaries as follows. First, she read the original full paper and wrote down a small list of the main ideas contained in that paper. Second, she read each summary in turn, without knowing which system produced the summary, and then ranked the three summaries according to 2 criteria:

- The summary must captures the main ideas of the source text.
- The summary must be understandable for users that did not read the full-text, considering the main idea of the text.

The results and the selected papers are showed in the Table 2.

Table 2. Results

Paper	Best Summary	Second-Best Summary	Third-Best Summary
FT921-8562	CMU	Our System	Cornell
FT943-2627	Our System	CMU	Cornell
WSJ900914-0056	Cornell	Our System	CMU
WSJ900921-0017	Our System	Cornell	CMU
WSJ900928-0072	CMU	Our System	Cornell
WSJ910325-0154	CMU	Our System	Cornell
WSJ911002-0124	Our System	Cornell	CMU

Overall, both our system and the CMU system achieved good rankings – each of them was considered the best summary in three out of the seven papers. Both systems produced results better than the Cornell system.

However, recall that a direct comparison between our system and the other two ones is not entirely fair, since our system selects 10% of the number of lines, while the other two systems select 10% of the number of characters of the source text. Actually, the summaries produced by our system have, in average, 15% of the number of characters of the source text.

This experiment indicates that we have obtained some promising results, but we plan to make more tests in the future. The abstracts generated by our system for these seven documents can be downloaded from http://www.ppgia.pucpr.br/groups/textmining.

References

1. Cavnar, W. B. Using An N-Gram-Based Document Representation With a Vector Processing Retrieval Model. *Proc. TREC-3 (Third Text Retrieval Conf.)*. Gaithersburg, USA. 1994.
2. Hearst, Marti A. TextTiling: A Quantitative Approach to Discourse Segmentation. *Technical Report 93/24*. University of California, Berkeley. 1993.
3. Joachims, T. A Probabilistic Analysis of Rocchio Algorithm with TFIDF for Text Categorization. *Technical Report CMU-CS-96-118*. Dept of Comp. Sci., CMU. 1996.
4. Larocca Neto, Joel; Santos, Alexandre Denes dos; Kaestner, Celso A.; Freitas, Alex A. . A Text Mining Tool for Document Clustering and Text Summarization. *Proceedings of The Fourth International Conference on The Practical Application of Knowledge Discovery and Data Mining*, 41-56. Manchester, UK. Apr, 2000. The Practical Application Company Ltd.
5. Mani, I.; House, D.; Klein, G.; Hirschman, L.; Obrsl, L.; Firmin, T.; Chrzanowski, M.; Sundheim, B. *The TIPSTER SUMMAC Text Summarization Evaluation.* MITRE Technical Report MTR 98W0000138. The MITRE Corporation, Oct. 1998.
6. Manning, C. D. ; Schutze, H. . Foundations of Statistical Nat. Lang. Proc. MIT Press. 1999.
7. Porter, M.F. An algorithm for suffix stripping. *Program 14*, 130-137. 1980. Reprinted in: Sparck Jones, K. and Willet, P. (Eds.) *Readings in Information Retrieval*, 313-316. Morgan Kaufmann, 1997.
8. Salton, G. and Buckley, C. Term-weighting approaches in automatic text retrieval. *Information Processing and Management 24,* 513-523. 1988. Reprinted in: Sparck Jones, K. and Willet, P. (Eds.) *Readings in Information Retrieval*, 323-328. Morgan Kaufmann, 1997.
9. Witten, Ian H.; Moffat, Alistair; Bell, Timothy C. Managing Gigabytes. *Van Nostrand Reinhold*. New York. 1994.