Summarization by Logic Segmentation and Text Entailment

Doina Tatar, Emma Tamaianu-Morita, Andreea Mihis and Dana Lupsa

University "Babes-Bolyai"
Cluj-Napoca
Romania
{dtatar,mihis,dana}@cs.ubbcluj.ro,etamaian@yahoo.com

Abstract. As the phenomenon of information overload grows day by day, systems that can automatically summarize documents become increasingly studied and used. This paper presents some original methods for text summarization of a single source document by extraction. The methods are based on some of our own text segmentation algorithms. We denote them logical segmentations because for all these methods the score of a sentence is the number of sentences of the text which are entailed by it. The first method of segmentation and summarization is called Logical TextTiling (LTT): for a sequence of sentences, the scores form a structure which indicates how the most important sentences alternate with ones less important and organizes the text according to its logical content. The second method, Pure Entailment uses definition of the relation of entailment between two texts. The third original method applies Dynamic Programming and centering theory to the sentences logically scored as above. The obtained ranked logical segments are used in the summarization. Our methods of segmentation and summarization are applied and evaluated against a manually realized segmentation and summarization of the same text by Donald Richie, "The Koan", [9]. The text is reproduced at [14].

1 Introduction

Systems that can automatically summarize documents become increasingly studied and used. As a summary is a shorter text (usually no longer than a half of the source text) that is produced from one or more original texts keeping the main ideas, the most important task of summarization is to identify the most informative (salient) parts of a text comparatively with the rest. Usually the salient parts are determined on the following assumptions [6]:

- they contain words that are used frequently;
- they contain words that are used in the title and headings;
- they are located at the beginning or end of sections;
- they use key phrases which emphasize the importance in text;
- they are the most highly connected with the other parts of text.

Received 24/10/07
Accepted 07/12/07
Final Version 12/01/08

If the first four characteristics are easy to achieve and verify, the last one is more difficult to establish. For example, the connectedness may be measured by the number of shared words, synonyms, anaphora [7],[8]. On the other hand, if the last assumption is fulfilled, the cohesion of the resulting summary is expected to be higher than if this one is missing. In this respect, our methods assure high cohesion for the obtained summary while the connectedness is measured by the number of logic entailments between sentences.

In this paper we present some methods for summarization of a single source document by extraction based on a previous segmentation. The methods are genre independent and application independent, and take advantage of discourse structure.

The paper is structured as follows: in Section 2, some notions about textual entailment are discussed. Some basics about segmentation of discourse and Logical TextTilling algorithm are given in Section 3, where some different variants of our methods are presented. Summarization by segmentation is the topic of Section 4. An original dynamic programming algorithm for segmentation and summarization is presented in Section 5. We have applied the above methods of segmentation and summarization to the text [9]. Section 6 summarizes the statistics of the results. We finish the article with conclusions and possible further work directions.

2 Text entailment

In [10] are presented three methods for recognizing the text entailment, based on the following directional principle: A text T entails an hypothesis H, denoted by $T \rightarrow H$, iff H is less informative than T. The first method in [10] is obtained from the text-to-text metric introduced in [1] and from the lexical resolution introduced in [11] and consists in the verification of the relation: $sim(T,H)_T \leq sim(T,H)_H$. Here $sim(T,H)_T$ and $sim(T,H)_H$ are text-to-text similarities introduced in [1]. The second method in [10] calculates the similarity between T and H by *cosine*, the third by a variant of Levenshtein distance. All these methods confirm the above principle, the highest computational precision being realized by the second method. We use in this paper the second method.

A very simple algorithm of segmentation (denoted Pure Entailment) based on the entailment relation is the following:

```
Input: Text= $\{S_1, S_2, \cdots S_n\}$
Output: Segments $Seg_1, ..., Seg_t$.
    $k = 2, t = 1, Seg_t = \{S_1\}$
    while $k < n$ do
      if $(Seg_t \rightarrow S_k)$
       then
          $Seg_t := Seg_t \cup \{S_k\}$
       else
          $t := t + 1, Seg_t := \{S_k\}$
     endif
     k:=k+1
```

```
        endwhile
```

A method of summarization based on the entailment relation is:

```
Input: Text= $\{S_1, S_2, \cdots S_n\}$
Output: Summary $S$.
    $S = \{S_1\}; i = 2$
    while $i < n$ do
        if not $(S \to S_i)$
          then
              $S := S \cup \{S_i\}$
        endif
        i:=i+1
    endwhile
```

3 Segmentation by Logical TextTiling

To observe the difference between the algorithm to detect subtopic structure (as TextTiling [3]) and our algorithm oriented to detect logical structure of a text, let us emphasize that for us, the unit of discourse is a sentence, with a level of importance which is different from the level of importance of its neighbors. A logical segment is a contiguous piece of text (a sequence of sentences) that is linked internally but disconnected from the adjacent text. This feature assures the local coherence of a segment.

The main differences between TextTiling (TT) and our Logical TextTiling (LTT) method are:

- The tokenization in TT is made by dividing the text into token-sequences (pseudosentences) of a predefined size. In LTT the individual units are sentences;
- In TT a score is assigned to each token-sequence boundary i, calculating how similar the token-sequences $i - k$ through i are to token-sequence from $i + 1$ to $i - k + 1$.
  In LLT the score of a sentence S_i, $score(S_i)$, is the number of sentences of the text which are entailed by S_i (or as in section 3.1). A higher score of a sentence denotes a higher importance. This fact is in accordance with the following text entailment criteria: A text T entails a text H iff the text H is less informative than T [10]. So, the more sentences are entailed by a sentence S_i, the higher its importance is.

A boundary (a shift in discourse or a gap) in LTT is determined by a change in the configuration of logical importance of sentences (see Fig.1. and Fig.2.). In such a way we have a structural basis for dividing up the discourse into a series of smaller segments, each with a different configuration of the importance of components.

The obtained logical segments could be used effectively in summarization. In this respect, our method of summarization falls in the discourse-based category. In contrast with other theories about discourse segmentation, as Rhetorical

Structure Theory (RST) of Mann and Thompson (1988), attentional / intentional structure of Grosz and Sidner (1986) or parsed RST tree of Daniel Marcu (1996), our Logical TextTiling method (and also TextTiling method [3]) supposes a linear segmentation (versus hierarchic segmentation) which results in an advantage from a computational viewpoint.

The algorithm LTT of segmentation is :

```
INPUT: A sequence of sentences $S_1, ...S_n$ and a list of scores $score(S_1), ...score(S_n)$
OUTPUT: A list of segments $Seg_1, ...Seg_j$
    $j = 1; p = 1; Seg_1 = \{S_1\}$
    dir=up;
    while $p < n$ do
        if $score(S_{p+1}) > score(S_p)$
            then
                if dir = up
                    then
                        $Seg_j = Seg_j \cup \{S_{p+1}\}$
                    else
                        $j = j + 1$
                        $Seg_j = \{S_{p+1}\}$
                        dir=up
                endif
            else
                $Seg_j = Seg_j \cup \{S_{p+1}\}$
                dir = down
        endif
        $p = p + 1$
    endwhile
```

A very useful variant of the algorithm is to begin a new segment only if the score $score(S_p)$ is bigger than a given threshold.

3.1 ArcInt and ArcReal scores

As variants of the above score of a sentence as the number of entailed sentences we introduced in this paper two different types of scores: ArcInt and ArcReal. Because the entailment relationship takes place between two sentences, we will consider that those two sentences are bounded together by an "arc". From a single sentence point of view, it can be the starting point of such an "arc", or it can be the destination point, or an in-going arc can pass over it. If i<k<j and sentence i entailes sentence j, then over sentence k passes an "arc" that starts from i and goes to j. If all these "arcs" that start from a sentence, that stop in that sentence and that pass over it are counted, the ArcInt score is obtained. In a way, it shows how the current sentence is linked to the other sentences. But because an entailment relationship is stronger if the sentences involved are close to one another, the sum of all arcs over their length could give a more

precise measure: ArcReal. The following formulas specify more clearly ArcInt and ArcReal measures for a sentence k:

$$ArcInt_k = \sum_{i=1}^{k} \sum_{j=k,i<>j}^{n} entails_{i,j} + \sum_{i=k}^{n} \sum_{j=1,i<>j}^{k} entails_{i,j}$$

$$ArcReal_k = \sum_{i=1}^{k} \sum_{j=k,i<>j}^{n} \frac{entails_{i,j}}{\mid i-j \mid} + \sum_{i=k}^{n} \sum_{j=1,i<>j}^{k} \frac{entails_{i,j}}{\mid i-j \mid}$$

where $entails_{i,j} = 1$ if $s_i \rightarrow s_j$ and $entails_{i,j} = 0, otherwise$.

The application of LTT method to sentences scored by ArcInt and ArcReal we will denote in the paper by ArcInt and ArcReal methods. These three methods: LTT, ArcInt and ArcReal are denoted as Logical methods.

4 Summarization by segmentation

The representation of the text as a structure is motivated by our goal of generating summary extracts, where a salience function must tell us which sentences are more salient, to be introduced in the summary. However, given a set of segments we need a criterion to select the segments and those sentences from a segment which will be introduced in the summary. The summarization algorithm from a sequence of segments is the following:

```
INPUT: The segments $Seg_1, ...Seg_j$, the length of summary X (as parameter);
OUTPUT: A summary $SUM$ of length X .

Calculate the "salience" of each segment and rank the segments in $Seg_{i_1}, ...Seg_{i_j}$;
Calculate the number of sentences in each segment $Seg_{i_s}$, $c_{i_s}$;
Select first $k(< j)$ segments such that $\sum_{Seg_{i_s}} c_{i_s} = X$;
Reorder selected k segments by their occurrence in text: $Seg'_1, ...Seg'_k$ ;
$SUM = \{Seg'_1, ...Seg'_k\}$ ;
```

In our paper we considered equal salience of segments and we selected from each segment as the most salient sentence, the sentence with the maximal score.

5 Dynamic Programming algorithm

In this section we describe our method of summarization by Dynamic Programming. The segmentation of the text is made from the summary, choosing as the first sentence of a new segment a sentence from the summary. In order to meet the coherence of the summary the algorithm selects the chain of sentences with the property that two consecutive sentences have at least one common word. This corresponds to the continuity principle in the centering theory which requires that two consecutive units of discourse have at least one entity in common ([7]). We make the assumption that, with each sentence, is associated a score that reflects how representative is that sentence. In these experiments the scores are the logical scores as used in the previous sections. The score of a selected summary is the sum of individual scores of contained sentences. The summary will be selected such that its score is maximum.

The idea of the Dynamic Programming Algorithm is the following.

Let us consider that δ_i^k is the score of the summary with length k that begins with the sentence S_i. Suppose that we ignore for the moment the requirement that each two consecutive sentences in the summary must have at least one common word. Then the next relation holds:

$$\delta_i^k = score(S_i) + max_{j\ (i<j)}\ (\delta_j^{k-1}\ )$$

If we introduce a penalty for the case in which two consecutive sentences have no words in common, the recursive formula becomes: $\delta_i^k = max_{j\ (i<j)}(score(S_i) + \delta_j^{k-1}\ )$ if S_i and S_j have common words and $\delta_i^k = max_{j\ (i<j)}(penalty*(score(S_i)$ $\delta_j^{k-1}\ ))$ if S_i and S_j have no common words. When running the implementation of the algorithm we used a penalty with value equal to 1/10.

Dynamic Programming Algorithm

INPUT:

-the text to be sumarized, that is a sequence of sentences $S_1, ..., S_n$

-$score(S_i)$ – scores associated to each sentence

-the length X of the summary

OUTPUT:

A summary SUM of length X

```
for $i = n$ downto 1 do
  $\delta_i^1 = score(S_i)$
  for $k = 2$ to $X$ do
    $\delta_i^k = max_{j\ (i<j)}(score(S_i) + \ (\delta_j^{k-1}\ ))$ if $S_i$ and $S_j$have common words
    $\delta_i^k = max_{j\ (i<j)}(penalty * (score(S_i) + \ (\delta_j^{k-1}\ )))$ otherwise
    $h_i^k = argmax_{j\ (i<j)}(score(S_i) + \ (\delta_j^{k-1}\ ))$ if $S_i$ and $S_j$ have common words
    $h_i^k = argmax_{j\ (i<j)}(penalty * (scor(S_i) + \ (\delta_j^{k-1}\ )))$ otherwise
  endfor
endfor
$i = argmax_j(\ \delta_j^X\ )$
$SUM = empty$
$for\ k = X, 1, step = -1\ do$
    $SUM = SUM \cup \{S_i\}$
    $i = h_i^k$
endfor
```

A short comparison with greedy algorithm in [7] is the following: our algorithm is based on previously calculated scores for sentences while greedy algorithm starts with previously calculated scores for sentences, but continuously adjusts them, depending on the sentences chosen in the summary. A second aspect is that our algorithm obtains the optimum solution. Because the scores of the sentences in greedy algorithm depend on the obtained summary, it would be difficult to have in this case a non-backtraking algorithm that obtains the optimum solution.

6 Experiments

It is now generally accepted that for single news article systems produce summaries indistinguishable from those produced by humans [8]. However, we apply our algorithm of segmentation and summarization to the narrative literary text by Donald Richie "The Koan" [9] reproduced at http:// www.cs.ubbcluj.ro/ dtatar/ nlp/ Koan-fara-anaph.txt. The manual anaphora resolution is reproduced at http:// www.cs.ubbcluj.ro/ dtatar/ nlp/ anaphEmma.txt. The structures of logically scored sentences for initial text and for anaphora solved text are presented in Fig.1. The structure of ArcInt and ArcReal scored sentences is presented in Fig.2.

6.1 Evaluation of segmentation

There are several ways to evaluate a segmentation algorithm. These include comparing the segments against that of some human judges, comparing the segments against other automated segmentation strategies and, finally, studying how well the results improve a computational task [7]. We will use all these ways of evaluation, including the study how our segmentation method effect the outcome of summarization (Section 6).

Regarding the comparison with the human judge, the method is evaluated according to [3]:

- how many of the same (or very close) boundaries (gaps) with the human judge the method selects out of the total selected (precision);
- how many true boundaries are found out of the total possible (recall)

We compared with Short (19 segments) and Long (22 segments) manual segmentations (as the gold standards) the results of segmentation obtained by the methods: LLT (with the variants of ArcInt and ArcReal scoring, with and without anaphora solved) , Pure Entailment (with and without anaphora solved), Dynamic Programming (with and without resources, with and without anaphora solved). The resources used in Dynamic Programming method are enumerated in Section 6.3.

The evaluation of the segmentation against the manual summaries ($Manual = Short, Long$) is made with the following measures:

$$Precision_{Method,Manual} = \frac{Number\, of\, correct\, gaps}{Number\, of\, gaps\, of\, Method}$$

$$Recall_{Method,Manual} = \frac{Number\, of\, correct\, gaps}{Number\, of\, gaps\, of\, Manual}$$

where $Number\, of\, correct\, gaps$ is the numbers of begins of segments found by $Method$ which differ with -1,0,+1 to the begins of segments found by $Manual$.

The results are presented in the following table:

Table 1. The results of manual segmentation in Short and Long variants. The results of LTT and PE methods

Manual Short		Manual Long		Method LLT		Method PE	
Segment	Sentences	Segment	Sentences	Segment	Senteces	Segment	Sentences
Seg.1	1	*Seg*.1	1	*Seg*.1	1 – 2	*Seg*.1	1
Seg.2	2 – 6	*Seg*.2	2 – 4	*Seg*.2	3 – 5	*Seg*.2	2 – 6
Seg.3	7 – 9	*Seg*.3	5 – 6	*Seg*.3	6 – 7	*Seg*.3	7 – 8
Seg.4	10 – 12	*Seg*.4	7 – 9	*Seg*.4	8 – 11	*Seg*.4	9
Seg.5	13 – 14	*Seg*.5	10 – 12	*Seg*.5	12 – 14	*Seg*.5	10
Seg.6	14 – 15	*Seg*.6	13 – 14	*Seg*.6	15 – 17	*Seg*.6	11
Seg.7	16 – 20	*Seg*.7	15 – 18	*Seg*.7	18 – 23	*Seg*.7	12
Seg.8	21 – 24	*Seg*.8	19 – 20	*Seg*.8	24 – 28	*Seg*.8	13
Seg.9	25 – 29	*Seg*.9	21 – 22	*Seg*.9	29 – 32	*Seg*.9	14 – 16
Seg.10	30 – 32	*Seg*.10	25 – 29	*Seg*.10	33 – 35	*Seg*.10	17
Seg.11	33	*Seg*.11	30 – 32	*Seg*.11	36 – 39	*Seg*.11	18 – 20
Seg.12	34 – 38	*Seg*.12	33	*Seg*.12	40 – 43	*Seg*.12	21
Seg.13	39 – 43	*Seg*.13	34 – 38	*Seg*.13	44 – 46	*Seg*.13	22
Seg.14	44 – 45	*Seg*.14	39 – 43	*Seg*.14	47 – 49	*Seg*.14	23
Seg.15	46 – 50	*Seg*.15	44 – 45	*Seg*.15	50 – 51	*Seg*.15	24
Seg.16	51 – 52	*Seg*.16	46 – 50	*Seg*.16	52 – 55	*Seg*.16	25
Seg.17	53 – 55	*Seg*.17	51 – 52	*Seg*.17	56 – 58	*Seg*.17	26
Seg.18	56	*Seg*.18	53 – 55	*Seg*.18	59 – 60	*Seg*.18	27 – 32
Seg.19	57 – 65	*Seg*.19	56	*Seg*.19	61 – 65	*Seg*.19	33
		Seg.20	57 – 60			*Seg*.20	34
		Seg.21	61 – 63			*Seg*.21	35 – 41
		Seg.22	64 – 65			*Seg*.22	42
						...	...

To save space we omit the description of the segments 23 to 32 of PE methods.

The comparison between the Precision and the Recall of different logical methods presented in this paper reported to Short and Long manual segmentation is given in the Fig.3. and Fig.4..

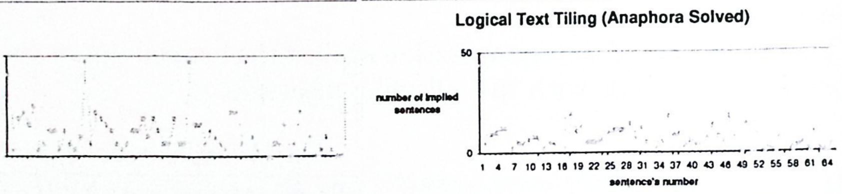

Fig. 1. The logical structure of the text.

The different results of Dynamic programming method are presented in Fig.5. and Fig.6..

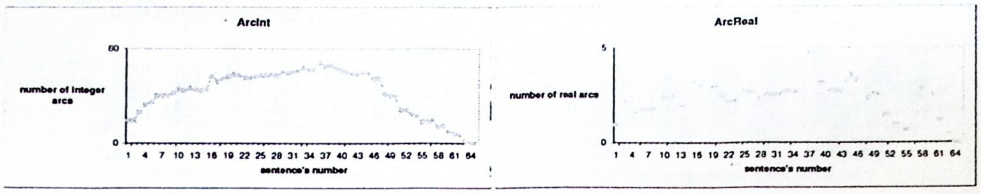

Fig. 2. ArcInt and ArcReal structure of the text.

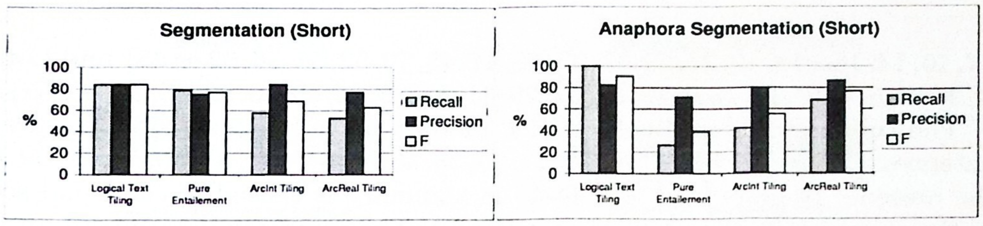

Fig. 3. Comparison of Logical methods with Short segmentation

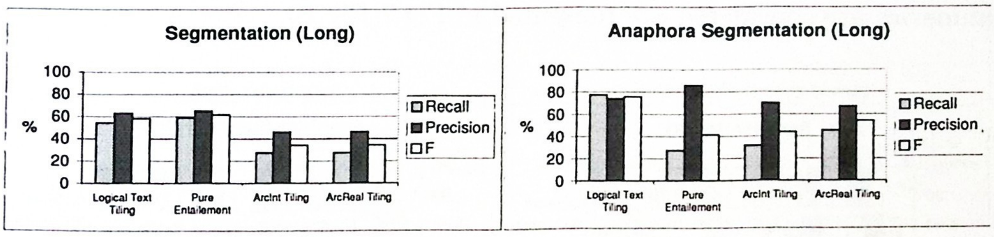

Fig. 4. Comparison of Logical methods with Long segmentation

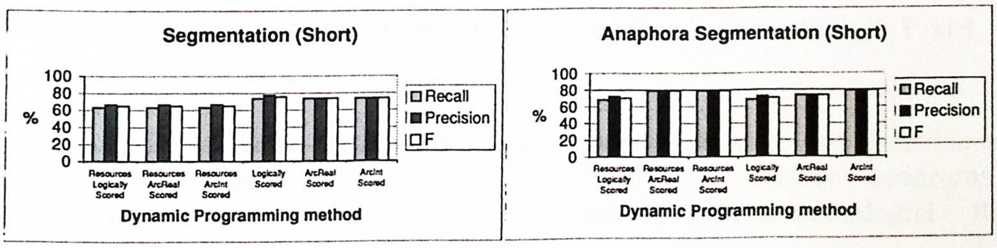

Fig. 5. Comparison of Dynamic Progr. methods with Short segmentation

6.2 Summarization

The gold standards for summaries are obtained manually from manually realized segmentations. The summaries are, in Short and Long variants: Short= $\{1, 2 +$

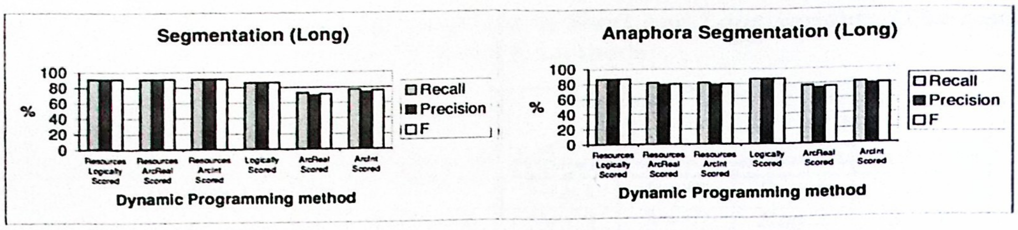

Fig. 6. Comparison of Dynamic Progr. methods with Long segmentation

$4, 7, 10, 14, 16 or 18, 19, 24, 29, 31, 33, 38, 41, 45, 50, 52, 55, 56, 62\ or\ 65\}$ and Long $=$ $\{1, 2+4, 6, 7, 10, 14, 16\ or\ 18, 19, 24, 29, 31, 33, 38, 41, 45, 50, 52, 55, 56, 62, 65\}$.

The summary determined by LTT method on initial text is formed by the sentences: $\{1, 4, 6, 9-10, 12, 16, 18, 27, 29, 33, 36, 40, 44-45, 47, 50, 54, 57, 59, 62\}$. The presence of pairs 9-10 and 44-45 in summary is caused of the equal scores for the sentences 9,10 and 44,45 (see Fig.1.). The summary determined by LTT method on the text with Anaphora solved is formed by the sentences: $\{1, 5-6, 8, 11, 14, 16, 18, 20, 29, 31, 33, 36, 38, 40, 44, 47, 50, 52, 54, 57, 60, 62, 65\}$.

The precision and the recall of the different methods when the manually summaries are considered are presented in Fig.7-Fig.10.

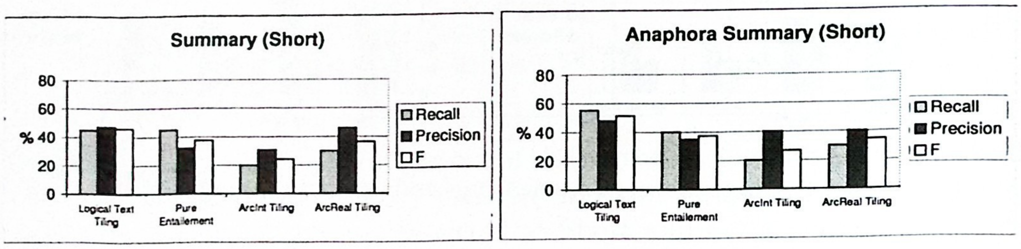

Fig. 7. Precision and Recall for Logical methods (Short summary as standard)

6.3 Implementation details

For Dynamic Programming algorithm we used the LYRL2004 stop-list, which is an ASCII file (english.stop), 3,589 bytes in size. It contains a list of 571 stop words, one per line, and was developed by the SMART project [5],[12] Also, to determine the common words between two sentences we used the files of nouns, adverbs, verbs and adjective of WordNet. To compare words in text with the words in these lists we used Porter stemmer. We used also a part of OpenNLP tools to identify sentences in text, and the tokens (words) at [13]. OpenNLP defines a set of Java interfaces and implements some basic infrastructure for

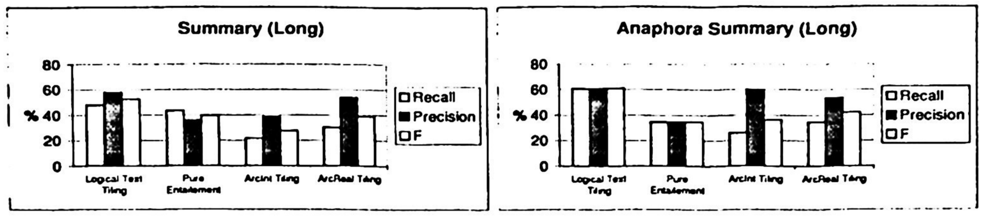

Fig. 8. Precision and Recall for Logical methods (Long summary as standard)

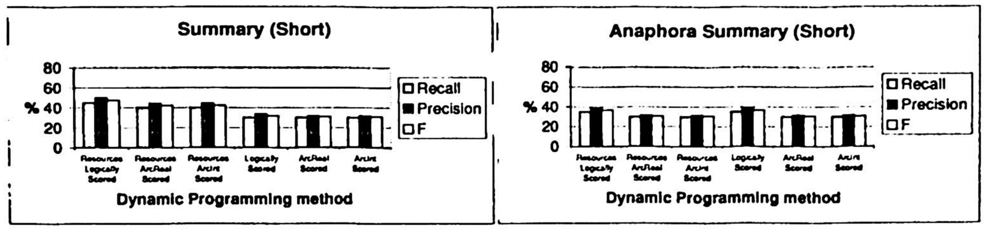

Fig. 9. Precision and Recall for Dynamic Programming method (Short summary as standard)

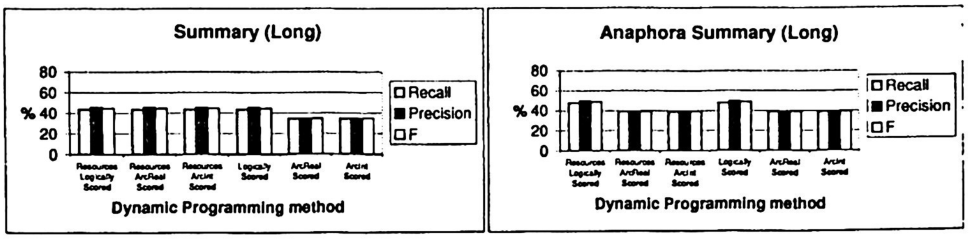

Fig. 10. Precision and Recall for Dynamic Programming method (Long summary as standard)

NLP components. We selected from there sentence detection and tokenization. Also, some of our algorithms have been realized as spreadsheet programs in Microsoft Excel.

7 Conclusion

As the abstraction is harder to be developed in summarization systems, the effort in extraction should be done with a view to increasingly coherent summaries with more accurate ways of identifying the important pieces of information. This paper shows that the discourse segmentation by text entailment relation between sentences is a good basis for obtaining highly accurate summaries (ranging from

30 per cent to over 60 per cent [4]) . Also, scoring the sentences on a logical base can give good results with a neutral method such as Dynamic Programming.

The algorithms described here are fully implemented and use text entailment between sentences without requiring thesaural relations or knowledge bases. The evaluation indices acceptable performance when compared against human judgement of segmentation and summarization. However, our method for computing the logical score of a sentence has the potential to be improved. We intend to improve our text entailment verification tool and to study how the calculation of logical score of a sentence (considering only neighbors of a target sentence, or considering also the number of sentences which imply the target sentence, etc) effects the segmentation and the summarization.

Our method was tested only on narrative texts. We intend to extend the evaluation using other types of texts.

Acknowledgments This work has been supported by CNCSIS Grant 91-037/19.09.2007 CONTO.

References

1. C. Corley, R. Mihalcea: "Measuring the semantic similarity of texts", Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment, Ann Arbor, June, 2005, pages 13-18.
2. I. Dagan, O. Glickman and B. Magnini: The PASCAL Recognising Textual Entailment Challenge, Proceedings of the PASCAL Work- shop, 2005.
3. M. Hearst: "TextTiling: A Quantitative Approach to Discourse Segmentation", Technical Report 93/24, University of California, Berkeley, 1993. 1997
4. R. Mitkov (editor): "The Oxford Handbook of Computational Linguistics", Oxford University Press, 2003.
5. Lewis, D. D.; Yang, Y.; Rose, T.; and Li, F. " RCV1: A New Benchmark Collection for Text Categorization Research", Journal of Machine Learning Research, 5:361-397, 2004.
6. D. Marcu: "From discourse structure to text summaries" in Proceedings of the ACL/EACL '97 Workshop on Intelligent Scalable Text Summarization, pp 82-88, Madrid,Spain.
7. C. Orasan: "Comparative evaluation of modular automatic summarization systems using CAST", PhD Thesis, University of Wolverhampton, UK, 2006.
8. D. Radev, E. Hovy, K. McKeown : "Introduction to the Special Issues on Summarization", Computational Linguistics, Vol.28, 2002, pp 399-408.
9. Donald Richie: "The Koan", in Zen Inklings. Some Stories, Fables, Parables and Sermons, New York and Tokyo: Weatherhill, 1991, pp.25-27
10. D. Tatar, G. Serban, A. Mihis and R. Mihalcea: "Text Entailment as directional relation", Proceedings of CALP07 Workshop at RANLP2007, pp 53-58, Borovets, Bulgaria, September 21-23
11. D. Tătar, M. Frenţiu: "Textual inference by theorem proving and linguistic approach", Studia Universitatis "Babeş- Bolyai", Seria Informatics, 2006, nr 2, pages 31-41.
12. http://jmlr.csail.mit.edu/ papers/ volume5/ lewis04a/ a11-smart-stop-list/english.stop
13. http://opennlp.sourceforge.net/
14. http://www.cs.ubbcluj.ro/dtatar/nlp/Koan-fara-anaph.txt