Extracting Sentence Segments for Text Summarization: A Machine Learning Approach

Wesley T. Chuang[1,2] and Jihoon Yang[2]
[1]Computer Science Department, UCLA, Los Angeles, CA 90095, USA
yelsew@cs.ucla.edu
[2]HRL Laboratories, LLC, 3011 Malibu Canyon Road, Malibu, CA 90265, USA
{yelsew|yang}@wins.hrl.com

Abstract

With the proliferation of the Internet and the huge amount of data it transfers, text summarization is becoming more important. We present an approach to the design of an automatic text summarizer that generates a summary by extracting sentence segments. First, sentences are broken into segments by special cue markers. Each segment is represented by a set of predefined features (e.g. location of the segment, average term frequencies of the words occurring in the segment, number of title words in the segment, and the like). Then a supervised learning algorithm is used to train the summarizer to extract *important* sentence segments, based on the feature vector. Results of experiments on U.S. patents indicate that the performance of the proposed approach compares very favorably with other approaches (including Microsoft Word summarizer) in terms of precision, recall, and classification accuracy.

Keywords: text summarization, machine learning, sentence segment extraction

1 Introduction

With tons of information pouring in every day, text summaries are becoming essential. Instead of having to go through an entire text, people can understand a text quickly and easily by means of a concise summary. The title, abstract and key words, if provided, can convey the main ideas, but they are not always present in a document. Furthermore, they may not touch on the information that users need to know.

SIGIR 2000 7/00 Athens, Greece

In order to obtain a good summary automatically, we are faced with several challenges. The first challenge is the extent to which we must "understand" the chosen text. It is very difficult to understand a document without digesting the whole text. However, detailed parsing takes a considerable amount of time, and does not necessarily guarantee a good summary. In our approach we do not claim to generate a summary by *abstract* (after understanding the whole text), but rather, by *extract* (meaning we attempt to extract key segments out of the text). This summarization by extract will be good enough for a reader to understand the main idea of a document, though the quality (and understandability) might not be as good as a summary by abstract.

The second challenge relates to program design. A summary may be more or less skewed subjectively depending on which features or characteristics are used to generate it - occurrence of title words in sentences, key words in headings, etc. Moreover, some features will give stronger indications of general usefulness than others. In short, features all have a degree of *subjectivity* and a degree of *generality* when they are used to extract a summary by identifying key sentences and fragments, and ideally, we are looking for those selection features that are independent of the types of text and user. These features will be used to distinguish the important parts from the less essential parts of the text and to generate a good summary.

Against this background, we propose an approach to automatic text summarization by *sentence segment extraction* using machine learning algorithms. We perform a "shallow parsing" [6] by looking at special markers in order to determine the sentence segments. Special markers and their associated significance, rhetorical relations, discriminate one portion of text from another. We define a set of features (e.g. average term frequency, rhetorical relations, etc.) for each sentence segment. Then we convert these features into a vector representation and apply machine

learning algorithms in order to derive the rules or conditions by which we will generate a summary. Regardless of whether the features have a high degree of subjectivity or generality, they all have a role to play in indicating which sentence segments in the text are more likely to be chosen as summary material. For example, features that are used in classical summarization methods [4, 1] such as title words, location, term frequency, etc., bear high contingency for selection. Features that come from *Rhetorical Structure Theory* [5] such as antithesis, cause, circumstances, concession, etc., will also determine the significance of a sentence segment. As we shall find out, machine learning will report to us whether one feature is useful at all in looking for summary material based on achieving a good balance between subjectivity and generality of summarization.

The rest of this paper is organized as follows: Section 2 presents some of the previous work in text summarization from which we developed our ideas. Section 3 describes our approach to text summarization based on sentence segment extraction. Section 4 presents the results of experiments designed to evaluate the performance of our approach. Section 5 concludes with a summary and discussion of some directions for future research.

2 Related Work

Luhn [4] and Edmundson [1] proposed a simple approach to automatic text summarization. They considered features such as average term frequency, title words, sentence location, and bonus/stigma words in order to extract sentences to make a summary. Their approaches performed fairly well despite their simplicity, and have been the basis in the area of automatic text summarization. However, their approaches ignored the structural aspect of the text which may contain significant information for summarization.

As an attempt to exploit the structural aspect of text, Marcu [6] provided a corpus analysis when he created a tree, known as the *Rhetorical Structure Theory (RST) Tree*, for all the segments in the text. After the tree is created for each document, segments are laid out in a manner in which more important segments occupy the upper level of the tree nodes, whereas less important segments reside deeper in the tree. As a result, summarization can be carried out by performing cuts at various depths of the tree, producing summaries of various lengths. His approach, however, suffers from the complexity issue. Every time a new document needs to be considered, the process for constructing its RST tree is very costly, not to mention the scalability factor for large amounts of data. Furthermore, when the text does not contain many rhetorical relations, there is little telling which segment is more important than the others.

There is another group of researchers who generated a summary by sentence extraction using the aforementioned non-structured features (e.g. title words) [3, 11]. Their significant contribution is that they made use of machine learning algorithms to determine the sentences to be extracted. However, since a sentence is the basic unit of their consideration, the resulting summary may still be unnecessarily long. In addition, there is a lack of the features based on the structural aspects (e.g. rhetorical relations) of sentences.

Our approach combines aspects of all the previous work mentioned above. We generate a summary by extracting sentence segments to make the summary concise. And we represent each sentence segment which is determined based on cue phrases, by using a set of non-structured as well as structured features, to both of which machine learning algorithms are applied in order to derive rules for summarization.

3 Design of a Text Summarizer Based on Sentence Segment Extraction

There are three steps in the design of our system: segmentation of sentences, feature representation of segments, and training of the summarizer.

3.1 Sentence Segmentation

Our segmentation method is basically the same as Marcu's [6]. A sentence is segmented by a *cue phrase*. (See [6] for detailed descriptions of the cue phrases and the segmentation algorithm.) The basic idea behind this is to separate out units (i.e. sentence segments) that possibly convey independent meanings. For instance, we can generate two sentence segments "I love playing tennis" and "because it is exciting" from the topic sentence "I love playing tennis because it is exciting". The cue phrase used here is *because* which connects the two segments (i.e. main and subordinate clauses) with a *cause* relationship. The purpose of segmentation is to use sentence segments as a basic unit for summarization. The complexity of the segmentation process is $O(n)$, where n is the number of sentences.

Figure 1 shows the segmentation of a sample patent data source taken from the *U.S. Patent and Trade-*

mark Office [8]. Segments are bounded by the bracket "]" with an integer (segment ID) numbering their sequences. Words enclosed by curly braces "}" are called *comma parenthesis* [6], which are considered as additional information inside a segment, and will be thrown out during summarization.

```
[ This invention relates in general to database management
systems performed by computers, and in particular, to a
method and apparatus for accessing a relational database over
the Internet using macro language files. 1]
   [ With the fast growing popularity of the Internet and the
World Wide Web { ( also known as " WWW " or the " Web " ) },
2] [ there is also a fast growing demand for Web access to
databases. 3] [ However, it is especially difficult to use
relational database management system { ( RDBMS ) } software
with the Web. 4] [ One of the problems with using RDBMS
software on the Web is the lack of correspondence between the
protocols used to communicate in the Web with the protocols
used to communicate with RDBMS software. 5]
   [ For example, the Web operates using the HyperText
Transfer Protocol { ( HTTP ) } and the HyperText Markup
Language { ( HTML ) }. 6] [ This protocol and language
results in the communication and display of graphical
information that incorporates hyperlinks. 7] [ Hyperlinks are
network addresses that are embedded in a word, phrase, icon
or picture that are activated 8] [ when the user selects a
highlighted item displayed in the graphical information. 9]
[ HTTP is the protocol used by Web clients and Web servers to
communicate between themselves using these hyperlinks. 10] [
HTML is the language used by Web servers to create and
connect together documents that contain these hyperlinks. 11]
[ In contrast, most RDBMS software uses a Structured Query
Language { ( SQL ) } interface. 12] [ The SQL interface has
evolved into a standard language for RDBMS software and has
been adopted as such by both the American Nationals Standard
Organization { ( ANSI ) } and the International Standards
Organization { ( ISO ) }. 13]
   [ Thus, there is a need in the art for methods of
accessing RDBMS software across the Internet network, and
especially via the World Wide Web. 14] [ Further, there is a
need for simplified development environments for such
systems. 15]
   [ To overcome the limitations in the prior art described
above, and to overcome other limitations that will become
apparent upon reading and understanding the present
specification, 16] [ the present invention discloses a
method and apparatus for executing SQL queries in a
relational database management system via the Internet. 17]
[ In accordance with the present invention, Web users can
request information from RDBMS software via HTML input forms,
which request is then used to create an SQL statement for
execution by the RDBMS software. 18] [ The results output by
the RDBMS software are themselves transformed into HTML
format for presentation to the Web user. 19]
```

Figure 1: A patent data source and its segmentation

3.2 Feature Representation

The sentence segments need to be represented by a set of features. As described before, there are two kinds of features we consider: *structured* and *non-structured*. The former are related to the structure of the text (e.g. rhetorical relations), while the latter are not (e.g. title words).

3.2.1 Rhetorical Relations

Mann and Thompson noted in their *Rhetorical Structure Theory* that a sentence can be decomposed into segments, usually clauses [5]. In a complex sentence, with two clauses, the main segment is called a *nucleus*, and its subordinate segment is called a *satellite* and is connected to the main segment by some kind of rhetorical relation. Figure 2 illustrates two examples of rhetorical relations. There are many such signaled by different cue phrases (e.g. *because, but, if, however,* $\cdots$). Generally, when a rhetorical relation occurs, the nucleus is considered as a more important segment – and has more chance of being in the summary – than its satellite counterpart.

Using Marcu's discourse-marker-based hypothesizing algorithm [6], we discover rhetorical relations on the base level of segments. In other words, we obtain the rhetorical relations of a segment to another segment in a nearby region instead of taking all the combinations of segments recursively to generate the whole RST tree – a computation which is significantly expensive. The complexity for finding such base-level rhetorical relations is $O(n^2)$, where n is the number of sentences.

A rhetorical relation $r(\texttt{name}, \texttt{satellite}, \texttt{nucleus})$ shows that there exists a relation (whose type is `name`) between the `satellite` and `nucleus` segments. The hypothesizing algorithm finds rhetorical relations for all the segments in the following form:

$$HRR = \bigcap_{i=1}^{n} \bigoplus_{j=1}^{k_i} r_{ij}$$

$$= (r_{11} \oplus r_{12} \oplus \ldots r_{1k_1}) \cap (r_{21} \oplus r_{22} \oplus \ldots r_{2k_2})$$
$$\cap \ldots \cap (r_{n1} \oplus \ldots r_{nk_n})$$

where k_i is the maximum distance to the salient unit as defined in [6]. Figure 3 shows the rhetorical relations produced for the patent data in Figure 1.

3.2.2 Feature Vector

We collect a total of 23 features and generate the feature vector:

$$\vec{F} = < f_1, f_2, f_3, f_4, \cdots, f_{22}, f_{23} >$$

They $(f_1, \cdots, f_{23})$ are to indicate the features for every segment we obtain in the sentence segmentation process. Though all the features in the vector will be taken into consideration together in classification (of segments to be included in the summary), we divide the features into the following three groups for analysis:

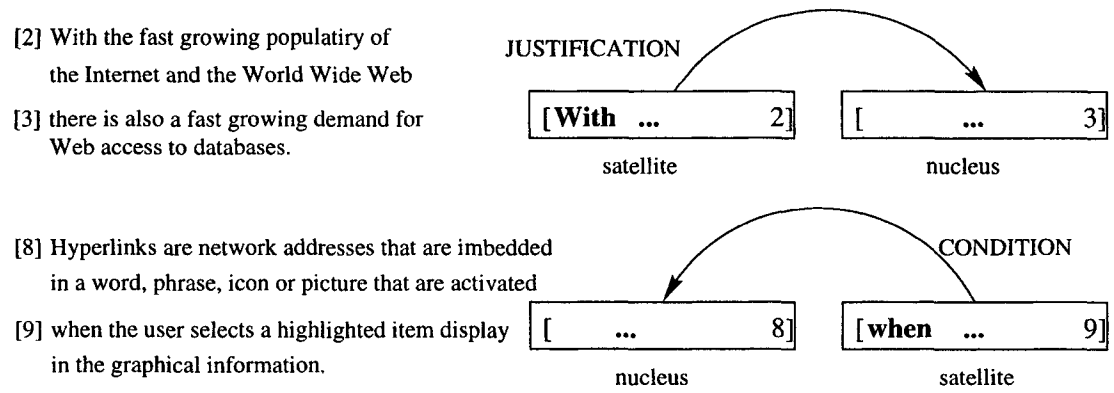

Figure 2: Sentence segments and their rhetorical relations

$$
HRR = \left\{ \begin{array}{l}
(r(JUSTIFICATION,2,3) \oplus r(JUSTIFICATION,1,3)) \cap \\
(r(ANTITHESIS,4,3) \oplus r(ANTITHESIS,5,3) \oplus r(ANTITHESIS,6,3) \oplus \\
\quad r(ANTITHESIS,4,2) \oplus r(ANTITHESIS,5,2) \oplus r(ANTITHESIS,6,2)) \cap \\
(r(EXAMPLE,6,5) \oplus r(EXAMPLE,7,5) \oplus r(EXAMPLE,8,5) \\
\quad \oplus r(EXAMPLE,6,4) \oplus r(EXAMPLE,7,4) \oplus r(EXAMPLE,8,4)) \cap \\
r(CONDITION,9,8) \cap \\
(r(JUSTIFICATION,13,14) \oplus r(JUSTIFICATION,13,15) \oplus \\
\quad r(JUSTIFICATION,12,14) \oplus r(JUSTIFICATION,12,15)) \cap \\
(r(PURPOSE,16,17) \oplus r(PURPOSE,16,18)) \cap \\
(r(ELABORATION,11,10) \oplus r(JOINT,10,12) \oplus r(JOINT,9,11) \oplus \\
\quad r(JOINT,9,12) \oplus r(ELABORATION,11,8) \oplus r(JOINT,8,12) \\
\quad \oplus r(ELABORATION,11,7) \oplus r(ELABORATION,12,7))
\end{array} \right.
$$

Figure 3: Rhetorical relations for the sample patent data

- Group I: 1. paragraph number, 2. offset in the paragraph, 3. number of bonus words, 4. number of title words, 5. average term frequency
- Group II: 6. antithesis, 7. cause, 8. circumstances, 9. concession, 10. condition, 11. contrast, 12. detail, 13. elaboration, 14. example, 15. justification, 16. means, 17. otherwise, 18. purpose, 19. reason, 20. summary relation
- Group III: 21. weight of nucleus, 22. weight of satellite, 23. max level.

Features 1-5 in Group I are non-structural attributes of the text. They are counters associated with the location of the segment, number of significant (bonus) words (out of the pre-defined set of 46 words) in the segment, and the average term frequency of the words in the segment.

Features 6-20 are distinct rhetorical relations. The features corresponding to the relations are initialized to 0's. When a segment is hypothesized with a relation, feature F_i for the relation will change its value by the following equation:

$$
F_i = \left\{ \begin{array}{ll} F_i + 1.0/x & \text{if nucleus} \\ F_i - 1.0/x & \text{if satellite} \end{array} \right.
$$

where x is the number of the asymmetric, exclusive-or relations being hypothesized with the segment. F_i shows how strong (in terms of its role as nuclei or satellites) the feature (relation) is for the corresponding segment.

Features 21-23 in the last group are collective descriptions of the rhetorical relations. For example, `weight of nucleus (satellite)` sums up all the occurrences in which a segment acts as a nucleus (satellite), regardless of which relation it possesses. `Max level` describes how many times, recursively, a segment can be a satellite of another satellite. The following semi-naive algorithm (see Figure 4) depicts how `max level` is determined. The algorithm has a complexity of $O(n^2)$, where n is the total number of relations. It is a simpler alternative in place of Marcu's complicated, expensive RST tree generation algorithm.

```
Input: Hypothesized rhetorical relations HRR
Output: The max level of each segment

// initialize the set of relation chains
RelationChain RC := null;

// get asymmetric relations from HRR
AsymmetricRhetoricalRelation ARR
   := asymmeticRelation(HRR);

// compute relation chains by checking all terms in
   all relations
for i := 1 to length(ARR)
   R_i := ARR(i);
   for j := 1 to length(R_i)
      r_ij = R_i(j);
      for k := 1 (k!=i) to length(ARR)
         R_k := ARR(k);
         for l := 1 to length(R_k)
            r_kl = R_k(l);
            if (r_ij.nucleus == r_kl.satellite)
               RC := RC ∪ concatenate(r_ij, r_kl);
            else if (r_ij.satellite == r_kl.nucleus)
               RC := RC ∪ concatenate(r_kl, r_ij);

// check the length of relation chains
countLevels(RC);
```

Figure 4: Algorithm for finding the max level of a satellite

3.3 Summarizer Training

Here the goal is to select a few segments as a summary that can represent the original text. With the feature vectors generated in previous steps, we can easily apply machine learning algorithms to train a summarizer (i.e. supervised learning). We are interested in seeing whether programs can quickly learn from our model summary and categorize which segments should be in the target summary and which should not. We want them to learn this for all 23 aforementioned features that are deemed representative.

A variety of machine learning algorithms have been proposed in the literature [7, 12]. We chose the decision tree algorithm (C4.5) [9, 10], the naive Bayesian classifier (Bayesian) [7], and the inter-pattern distance-based constructive neural network learning algorithm (DistAl) [12] for our experiments.

3.3.1 Decision Trees

Decision tree algorithms [9, 10] are one of the most widely used inductive learning methods. Among the various decision tree learning algorithms, we chose the C4.5 algorithm [10] to train the summarizer. A decision tree is generated by finding a feature that yields the maximum information gain. A node is then generated with a set of rules corresponding to the feature. This process is repeated for other features in succession until no further information gain is obtainable. In testing, a pattern is repeatedly compared with a node of a decision tree starting from the root and following appropriate branches based on the condition and feature value until a terminal node is reached. The pattern is then presumed to belong to the class the terminal node represents. C4.5 has been known to be a very fast and efficient algorithm with good generalization capability. (See [7] for detailed descriptions of the algorithm and examples.)

3.3.2 Naive Bayesian Classifier

We apply the naive Bayesian classifier as used in [3]:

$$P(c \in C \mid F_1, F_2, \ldots F_k)$$

$$= \frac{\Pi_{j=1}^{k} P(F_j \mid c \in C) P(c \in C)}{\Pi_{j=1}^{k} P(F_j)}$$

where C is the set of target classes (i.e. in the summary or not in the summary) and F is the set of features. That is, we are trying to find a class C that will have the highest probability of observing F. In our experiment, since the values of most of the features are *real* numbers, we assume a *normal distribution* for every feature, and use the *normal distribution density function* to calculate the probability $P(F_j)$. That is,

$$P(F_j) = \frac{1}{\sigma_j \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{F_j - \mu_j}{\sigma_j}\right)^2}$$

where μ_j and σ_j are mean and standard deviation for feature F_j, respectively.

3.3.3 DistAl

DistAl [12] is a simple and relatively fast constructive neural network learning algorithm for pattern classification. (A constructive learning algorithm builds a network dynamically by adding neurons as necessary (e.g. to obtain better classification accuracy) instead of using a fixed network architecture determined a priori. See [2] for general information on constructive learning.) The key idea behind DistAl is to add *hyperspherical* hidden neurons one at a time based on a

greedy strategy which ensures that each hidden neuron that is added correctly classifies a maximal subset of training patterns belonging to a single class. Correctly classified examples can then be eliminated from further consideration. The process is repeated until the network correctly classifies the entire training set. When this happens, the training set becomes linearly separable in the transformed space defined by the hidden neurons. In fact, it is possible to set the weights on the hidden-to-output neuron connections without going through an iterative, time-consuming process. It is straightforward to show that DistAl is guaranteed to converge to 100% classification accuracy on any finite training set in time that is polynomial in the number of training patterns. Moreover, experiments reported in [12] show that DistAl, despite its simplicity, yields classifiers that compare quite favorably with those generated using more sophisticated (and substantially more computationally demanding) learning algorithms.

4 Experiments

In our experiments, first, we used the three learning algorithms (C4.5, Bayesian, DistAl) and evaluated their performance. Next, we compared this to the performance of the summarizer in Microsoft Word, since it is one of the most popular softwares being used nowadays.

Finally, we compared both of these with the performance of a very simple heuristic based on predetermined weights for several features introduced before. These weights we used to assign each segment a score and to generate a summary containing only segments that had high scores. The score for each segment was based on this function:

$$Score(f, t, b, N, S) = \mu f + \alpha t + \beta b + \gamma N + \eta S$$

where f=average term frequency, t=number of title words, b=number of bonus words, N=number of times it acts as a nucleus, S=number of times it acts as a satellite. $\mu, \alpha, \beta, \gamma$, and η are arbitrary constants, with the values $0.05, 1.6, 1.4, 1.0$, and -0.5 assigned to them respectively in this experiment. *Score* gives all segments in a text a total ordering, but for fair comparison, we only selected the top n scoring segments, with n equal to the number of segments in the manual model summary.

4.1 Dataset

There exist huge amounts of text data on the Internet, and all of the texts can be subjected to automatic summarization under our approach. Among the varieties of data available, we chose some of the U.S. patent data for our experiments. This was because the patent data is of particular interest to many people including lawyers, inventors, and researchers who need to read through a huge amount of data rapidly to grasp a knowledge of the current status of an area.

We selected and used nine U.S. patents. However, instead of considering the entire huge patent descriptions, we used only the sections "*background of invention*" and "*summary of invention.*" This was to check the feasibility of our approach without spending too much time in data preparation. For each patent data source, we manually generated a model summary (by consensus among three people) to be used in training and evaluating the summarizer. Table 1 displays the total number of sentences, total number of segments, and the number of segments in the model summary. As can be seen, the size of the data sources (i.e. number of sentence segments) was reasonably big even though only a small number of patents and only specific sections of patents were considered.

Table 1: Patent dataset

ID	*sentences*	*segments*	*model summary segments*
1	58	75	25
2	29	33	14
3	36	48	16
4	45	77	20
5	16	19	5
6	95	139	17
7	76	98	25
8	23	29	6
9	30	39	11

4.2 Experimental Results

The performance of the summarizer is evaluated by a 9-fold cross-validation using the patent data in Table 1. In other words, eight patents are used for training the summarizer (in case learning is required, as in C4.5, Bayesian, and DistAl), and the remaining one patent is used for testing. This is repeated nine times using a different patent for testing each time.

We evaluate the results of summarization by classification *accuracy* as well as by *precision* and *recall.* Consider the following four different cases of relationships between the desired classes and the predicted classes:

	selected	*not selected*
in model summary	a	c
not in model summary	b	d

Accuracy concerns the percentage from which the classifier correctly classifies the summary:

$$Accuracy = \frac{a+d}{a+b+c+d}$$

$$= \frac{\#\ of\ segments\ correctly\ categorized}{total\ \#\ of\ segments}$$

Therefore, segments not in the summary and not selected are considered as correctly categorized, just as are segments in the summary which are correctly selected.

Precision and recall are slightly different and are defined as follows:

$$Recall = \frac{a}{a+c}$$

$$= \frac{\#\ of\ segments\ in\ model\ summary\ selected}{\#\ of\ segments\ in\ model\ summary}$$

$$Precision = \frac{a}{a+b}$$

$$= \frac{\#\ of\ segments\ in\ model\ summary\ selected}{\#\ of\ segments\ selected\ as\ summary}$$

Table 2 displays the performance of all the methods considered. As we can see from Table 2, all the three approaches using machine learning outperformed other approaches without learning. In particular, the DistAI and Bayesian classifiers show significant improvement over other approaches.

One interesting result is that the Microsoft Word summarizer produced the worst performance. Even the simple heuristic-based approach generated a better summary than the Microsoft Word summarizer. We do not know exactly the underlying mechanism that Microsoft Word uses to summarize a document. However, it appears that many co-occurring words are simply selected as summary cues. Therefore, the summary is mostly composed of incoherent fragments from the sentences.

5 Summary and Discussion

The design of an automatic text summarizer is of great importance in the current world which is so filled with data. It would reduce the pain people suffer reading huge amounts of data by offering them a concise summary for each document. We have developed an automatic text summarizer based on sentence segment extraction. It generates a summary based on the rules derived from *any* supervised machine learning algorithm. Our approach also has a polynomial time complexity.

The experimental results demonstrate the feasibility of our approach. Even the simple heuristic-based approach produced reasonable performance. (See Figures 5 and 6 for the snapshots of sentence segmentation and summarization.) All the three learning algorithms we considered successfully completed the task by generating a reasonable summary and their performance was better than that of the commercial Microsoft Word summarizer. In particular, our system with the DistAI and Bayesian learning algorithms outperformed the other approaches.

Some avenues for future research include the following: The system can be extended to handle various types of data (e.g. Web pages, multimedia, etc.) and larger data sources (in terms of both the number of model-summary-generating documents used and their length); The system can be applied prior to datamining (e.g. classification) for efficiency and higher quality; The system can be bolstered with techniques such as anaphora resolution and summary by abstract to generate a more coherent summary. Some of these challenges are the focus of our ongoing research.

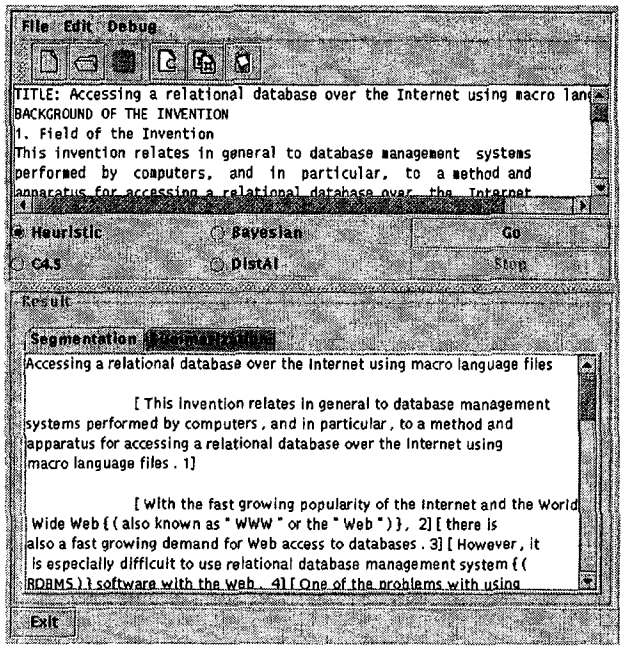

Figure 5: Snapshot of segmentation for the sample patent data

Table 2: Accuracy (a), precision (p) and recall (r) of different approaches in percents. *Avg* is the average and *Std* is the standard deviation of the 9-fold cross-validation.

	MS Word			Heuristic			C4.5			Bayesian			DistAI		
ID	a	p	r	a	p	r	a	p	r	a	p	r	a	p	r
1	62.7	46.4	50.0	73.3	60.0	60.0	70.7	32.0	61.5	72.0	60.0	48.0	72.0	44.4	48.0
2	48.5	40.0	42.8	63.6	57.1	57.1	60.6	21.4	60.0	60.6	52.9	64.3	57.6	40.0	28.6
3	33.3	23.5	25.0	62.5	43.8	43.8	64.6	37.5	46.1	60.4	41.2	43.8	72.9	45.5	31.3
4	64.9	33.3	35.0	74.0	50.0	50.0	88.3	60.0	92.3	75.3	52.6	50.0	76.6	47.6	50.0
5	57.4	28.6	40.0	78.9	60.0	60.0	68.4	60.0	42.9	63.2	37.5	60.0	84.2	60.0	60.0
6	76.3	5.6	5.9	77.0	5.9	5.9	82.7	29.4	29.4	84.2	40.0	58.8	87.8	28.9	64.7
7	61.2	25.9	28.0	71.4	44.0	44.0	70.4	32.0	40.0	79.6	60.9	56.0	74.5	33.3	40.0
8	75.9	40.0	33.3	72.4	33.3	33.3	75.8	71.4	71.4	51.7	50.0	71.4	100	100	100
9	66.7	40.0	36.4	69.2	45.5	45.5	71.8	9.1	50.0	76.9	100	18.2	77.0	54.5	54.5
Avg	60.8	31.6	32.9	71.4	44.3	44.3	72.6	39.0	54.8	69.3	55.0	52.3	78.1	50.5	53.0
Std	13.4	12.1	12.6	5.5	16.9	16.9	8.6	20.0	18.9	10.8	18.8	15.3	11.8	20.9	21.4

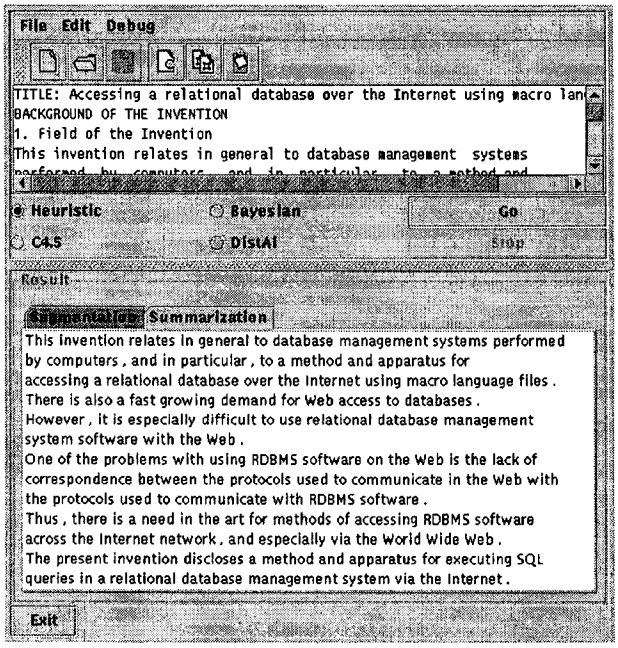

Figure 6: Snapshot of the heuristic-based summarization for the sample patent data

References

[1] H. Edmundson. New methods in automatic extracting. *Journal of the Association for Computing Machinery*, 16(2):264–285, 1969.

[2] V. Honavar and L Uhr. Generative learning structures for generalized connectionist networks. *Information Sciences*, 70(1-2):75–108, 1993.

[3] J. Kupiec, J. Pedersen, and F. Chen. A trainable document summarizer. In *Proceedings of the 18th ACM-SIGIR Conference*, pages 68–73, 1995.

[4] H. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165, 1958.

[5] W. Mann and S. Thompson. Rhetorical structure theory: Toward a functional theory of text. *Text*, 8(3):243–281, 1988.

[6] D. Marcu. *The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts.* PhD thesis, Department of Computer Science, University of Toronto, Toronto, Canada, 1997.

[7] T. Mitchell. *Machine Learning.* McGraw Hill, New York, 1997.

[8] T. Nguyen and V. Srinivasan. Accessing a relational database over the internet using macro language files, 1998. http://www.uspto.gov/.

[9] R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[10] R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[11] S. Teufel and M. Moens. Sentence extraction and rhetorical classification for flexible abstracts. In D. Radev and E. Hovy, editors, *Intelligent Text Summarization*, AAAI Spring Symposium, pages 16–25. AAAI Press, Menlo Park, CA, 1998.

[12] J. Yang, R. Parekh, and V. Honavar. DistAI: An inter-pattern distance-based constructive learning algorithm. *Intelligent Data Analysis*, 3:55–73, 1999.