Text Summarization as Controlled Search

Terry COPECK, Nathalie JAPKOWICZ, Stan SZPAKOWICZ
School of Information Technology & Engineering
University of Ottawa, Ontario, Canada
{terry, nat, szpak}@site.uottawa.ca

Abstract

We present a framework for text summarization based on the generate-and-test model. A large set of summaries is generated for all plausible values of six parameters that control a three-stage process that includes segmentation and keyphrase extraction, and a number of features that characterize the document. Quality is assessed by measuring the summaries against the abstract of the summarized document. The large number of summaries produced for our corpus dictates automated validation and fine-tuning of the summary generator. We use supervised machine learning to detect good and bad parameters. In particular, we identify parameters and ranges of their values within which the summary generator might be used with high reliability on documents for which no author's abstract exists.

1 Introduction

Text summarization consists in compressing a document into a smaller précis. Its goal is to include in that précis the most important facts in the document. Summarization can be performed by *constructing* a new document from elements not necessarily present in the original. Alternatively, it can be done by *extracting* from a text those elements, usually sentences, best suited for inclusion in a summary.

Our work takes the second approach. This paper describes a partially automated search engine for summary building. The engine has two components. A Summary Generator takes a text together with a number of parameters that regulate its operation in a variety of ways, and produces a summary of the text. A Generation Controller evaluates summaries of documents in a training corpus in order to determine the best parameters for use in generation.

Summary generation has been designed to combine a number of publicly-available text processing programs to perform each of its three main stages. Many parameter settings are possible; simply selecting among the programs available produces 18 possible combinations. Because many of the programs accept or require additional parameters themselves, evaluating the possible set of inputs manually is impossible.

Generation control is achieved through supervised machine learning that associates the summary generation parameters with a measure of the generated summary's quality. Training data consist of a set of features characterizing the document (Table 1) together with the parameter settings used to generate each summary and a rating of that summary. Features used to characterize a document are counts of its characters, words, sentences and paragraphs and other syntactic elements such as connectives and proper nouns. The number of one, two, three and four or more instances of bigrams, content bigrams and content phrases are also counted.

The summary rating is obtained by comparing it against the abstract, which we treat as the gold standard for the document on the grounds that a) the author of a document is well-suited to produce a summary of its contents; and b) for many documents the abstract is the only readily available item close to a summary. There are few alternatives.

While summary generation is fully automated, the controller was applied in a semi-manual way. We plan to automate the entire system in the near future.

FEATURE	DESCRIPTION	FEATURE	DESCRIPTION
chars	# of characters in the document	cphr3	# of three-instance content phrases
words	# of words in the document	cphr4	four-or-more instance content phrases
sents	# of sentences in the document	cbig	# of one-instance content bigrams
paras	# of paragraphs in the document	cbig2	# of two-instance content bigrams
kpiacnt	# of kpia instances in the document	cbig3	# of three-instance content bigrams
conncnt	# of (Marcu) connectives	cbig4	four-or-more instance content bigrams
pncnt	# of PNs, acronyms	abig	# of one-instance bigrams
contcnt	# of content phrases	abig2	# of two-instance bigrams
cphr	# of one-instance content phrases	abig3	# of three-instance bigrams
cphr2	# of two-instance content phrases	abig4	# four-or-more instance bigrams

Table 1: Document Features

2 Overall Architecture

The overall architecture of the system appears in Figure 1. The Summary Generator, shown in detail in Figure 2, takes the text to be summarized and one or more sets of parameter values and outputs one summary per set. These are then fed into the Generation Controller along with the parameter values used to produce them and features of the document summarized. The Generation Controller has two subcomponents. The Summary Assessor takes the input summary along with the document abstract and rates the summary with respect to the abstract. The Parameter Assessor takes the set of parameter value sets accumulated during summary generation together with ratings and uses the ratings to determine the best and worst combinations of parameters.

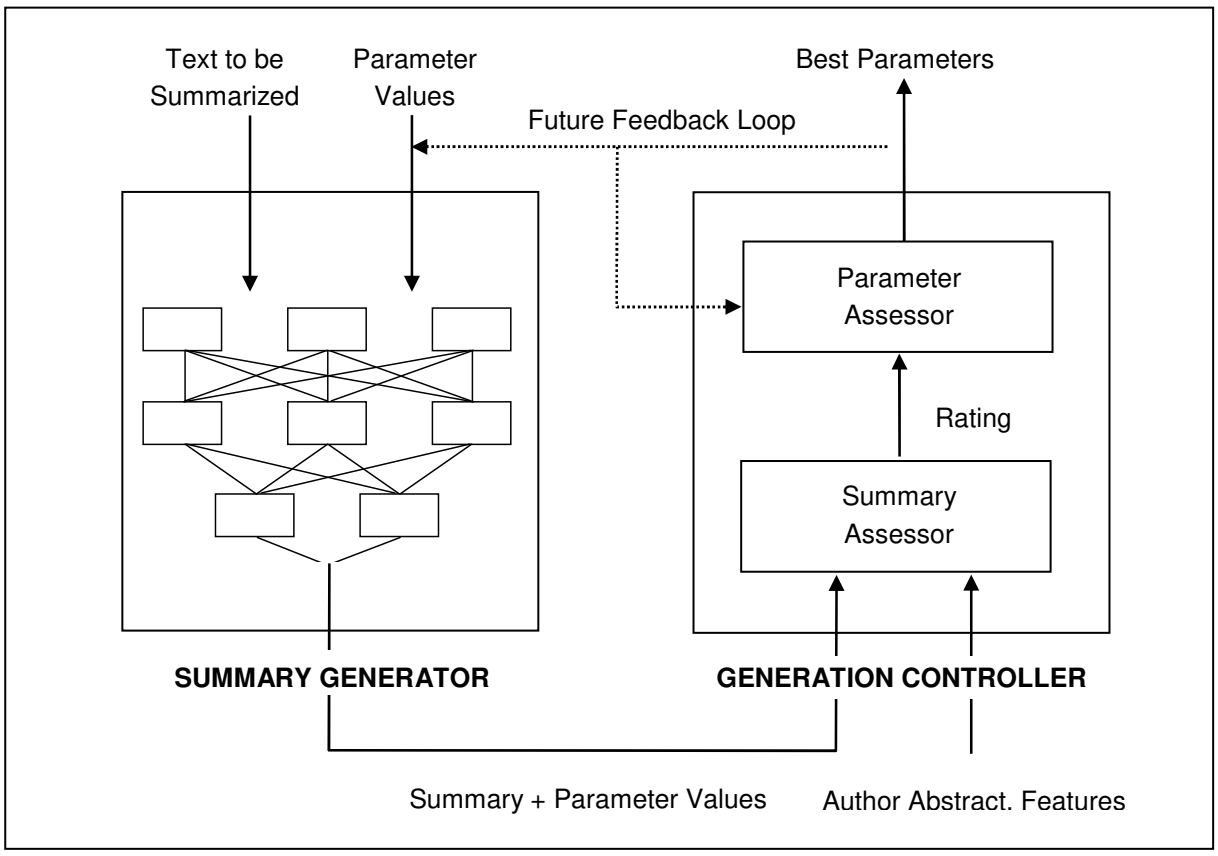

Figure 1. Overall System Architecture

3 The Summary Generator

We generate summaries on the assumption a summary composed of adjacent, thematically related sentences should convey more information than one whose sentences are chosen independently and likely bear no relation to their neighbours.

Let's first discuss generation methodology. A text to be summarized is *broken down* into a series of sequences of sentences or *segments* which talk about the same topic. Segments are then rated by *identifying keyphrases* in the document and ranking each on its overall number of *matches* for these phrases. The requested number of sentences is then selected from one or more of the top-ranked segments and presented to the user in document order. Lacking domain semantic knowledge or corpus statistics, document analysis based on segments and key phrases at least takes advantage of the statistically-motivated shallow semantics used in segmenters and keyphrase extractors.

To see if our general assumption holds regardless of the algorithm applied at each of the three main stages in summary generation, we designed a system with alternative modules for each stage; and where possible used for these tasks programs which, because they are in the public domain or made freely available, are well-known to the research community.

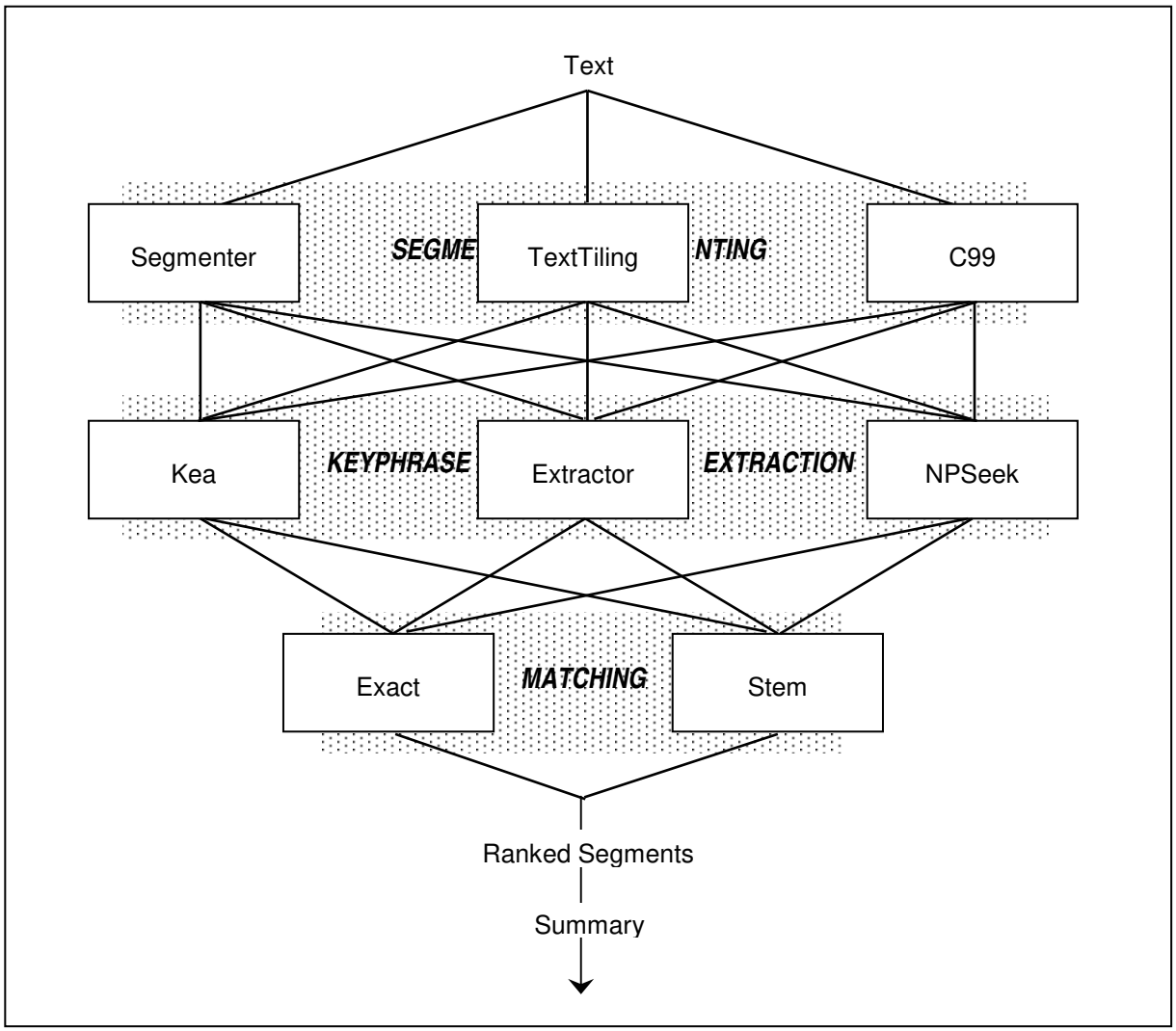

Figure 2. Summary Generator Architecture

Three text segmenters were used: Segmenter from Columbia University (Kan *et al.* 1998), Hearst's (1997) TextTiling program, and the C99 program (Choi 2000). Three programs were used to pick out keywords and keyphrases from the body of a text: Waikato University's Kea (Witten *et al.* 1999), Extractor from the Canadian NRC's Institute for Information Technology (Turney 2000), and NPSeek, a program developed at the University of Ottawa.

Once a list of keyphrases has been assembled, the next step is to count matches. The obvious way is to look for exact ones, but there are other ways, and arguments for using them. Consider the word *total*. It is used in *total up the bill*, *the total weight* and *compute a grand total*. Each matches exactly but is a different part of speech. Exact matching can be less exact than wanted. Should *totalling*, *totals* and *totalled*, all verbs, be considered to match *total*? If so, it is a small step to the case where the key, rather than the match, is suffixed—the token is *totalling* rather than *total*. Counting hits based on agreement between the root of a keyword and the root of a match is called *stem matching*. Kea contains modules which perform both exact and stem matching. We repackaged them as standalone programs. Figure 2 shows graphically how eighteen different combinations of programs can be invoked to summarize a text.

The particular segmenter, keyphrase extractor and type of matching to use are three of the parameters required to generate a summary. Three others must be specified: the number of sentences to include in the summary, the number of keyphrases to use in ranking sentences and segments, and the minimum number of matches to require in a segment. The Summary Generator constructs a summary by taking the most highly rated sentences from the segment with the greatest average number of keyphrases, so long as each sentence has at least the number of instances of keyphrases specified by *hitcount*. When this is no longer true, the generator moves to the next most highly rated segment, taking sentences until the required number has been accumulated.

4 The Generation Controller

4.1 Summary Assessment Component

Summary assessment requires establishing how effectively a summary represents the original document. This task is generally considered hard (Goldstein *et al*. 1999). Assessment is *highly subjective*: different people are likely to rate a summary differently. Summary assessment is also *goal-dependent*: how will the summary be used? More domain knowledge can be presumed in a specialized audience than in the general public and an effective summary must reflect this. Finally, and perhaps most important when many summaries are produced automatically, assessment is very *labour-intensive*: the assessor needs to read the full document and all summaries. It is difficult to recruit volunteers for such pursuits and very costly to hire assessors.

These considerations led us to focus on automating summary assessment. Although the procedure we settled on is fairly straightforward, the assumptions it is based on can be questioned. It is not clear whether the abstract can really serve as the gold standard for a summary. For example, we take abstracts in journal papers to be of high quality (Mittal *et al.* 1999). Yet in a corpus of 75 academic papers published in *Journal of Artificial Intelligence Research* during 1993-1996, we found 24% of abstract keyphrases do not occur in the body (std. dev. 13.3). We are however unable to find an alternative source for a large number of summaries.

The procedure is as follows: a list of keyphrases is constructed from each abstract, where a keyphrase is a sequence of tokens between stop words[1]. Such keyphrases are distinguished from those extracted during summarization by calling them *key phrases in abstract* or KPIAs. Summaries are rated on their number of KPIAs, with weight given both to the absolute number and to coverage of the total set

[1] A 980-word stop list was used, union of five publicly-available lists: Oracle 8 ConText, SMART, Hyperwave, and lists from the University of Kansas and Ohio State University.

of KPIAs. The rating procedure builds a set of KPIAs appearing in the summary, assigning a weight of 1.0 to each new KPIA added and 0.5 to each duplicating a KPIA already in the set. A summary's score is thus:

$$\sum 1.0 * KPIA_{unique} + 0.5 * KPIA_{duplicate}$$

This value is normalized for summary length and document KPIA coverage by dividing it by the total KPIAs in an equal number of sentences with the highest KPIA ratings.

The weighting of both KPIA count and coverage affects identification of the highest rated sentences in a text. In some circumstances a slightly higher overall score could be achieved by picking sentences with more unique and fewer absolute KPIA instances. Computing the highest possible score for a document running to hundreds of sentences is computationally quite expensive; it involves looking at all possible combinations of *sentcount* sentences. For normalization, simply totaling the counts for the sentences with the most KPIA instances to get a 'nearly best' score was deemed an adequate approximation.

4.2 Parameter Assessment Component

The Parameter Assessor identifies the best and worst outcomes from the parameter sets used in the Summary Generator so generation can be tuned to give best results. The task can be construed as supervised machine learning. Input is a vector of values describing each summary: the six parameters used to generate it, features characterizing the document summarized and the summary's KPIA rating. The parameters are:

- the *segmenter* (Segmenter, TextTiling, C99)
- the *keyphraser* (Extractor, NPSeek, Kea)
- the *match type* (exact or stemmed match)
- *key count*, the number of keyphrases used in generating the summary
- *sent count*, the number of sentences appearing in the summary
- *hit count*, the minimum number of matches for a sentence to be in the summary

Also included are twenty features listed in Table 1 that describe the document in question.

Output is a set of rules identifying which summary characteristics, including parameter settings, yield good or bad ratings. These rules tell us which values to use and which to avoid in future work with the same type of documents.

5 The Training Procedure

The C5.0 decision tree and rule classifier (Quinlan 1993) was chosen for parameter assessment because it uses state-of-the-art techniques and produces rules which, though often numerous, are individually easy to interpret. C5.0's decision trees can be used to predict likely values when these are missing. This proved crucial for the Document Understanding Conference, where documents without abstracts must be summarized and where KPIA values are therefore lacking.

Our procedure for DUC was to construct decision trees using data from the training corpus. Taking the summary rating as a value missing from the test corpus dataset, we used the trees to identify parameter settings for each document which maximized the absent value on a scale of five ('verybad' to 'verygood'). Settings that achieved this rating were then submitted to the summary generator to produce the 308 single document and 120 multi-document summaries sent to NIST. Multi-document summaries were generated by summarizing the single document produced by concatenating all files in a folder.

Ten-fold cross validation on the training data shows 16.3% classification error (std. dev. .01). Furthermore, the 5x5 confusion matrix indicates that 80% of these misclassifications were to the adjacent cell in the scale. This suggests the rules are reasonably trustworthy.

This assessment is borne out at least obliquely by the fact that the decision trees proved productive on the test corpus. When data describing the test documents and permuting all plausible parameter settings was submitted to the trees, 80% of documents had one or more

```
Rule 4: (9885/5433, lift 3.5)
        sents <= 53
        keyphrase = k
        keycount > 3
        sentcount > 5
        ->  class verygood  [0.450]

Rule 6: (6669/3452, lift 3.7)
        cphr <= 106
        abig3 > 0
        keyphrase = e
        ->  class verygood  [0.482]

Rule 12: (1257/386, lift 5.3)
        sents <= 53
        pncnt > 12
        cphr <= 269
        abig2 > 13
        abig2 <= 72
        keyphrase = e
        keycount > 5
        sentcount > 10
        hitcount <= 2
        ->  class verygood  [0.693]

Rule 1: (48415/21735, lift 2.1)
        keyphrase = n
        ->  class verybad  [0.551]

Rule 11: (1376/446, lift 2.6)
        sents <= 748
        paras > 248
        ->  class verybad  [0.676]

Rule 13: (755/126, lift 3.2)
        pncnt > 178
        cphr2 > 60
        cbig4 <= 9
        keycount <= 5
        hitcount > 2
        ->  class verybad  [0.832]

Rule 19: (516/190, lift 2.5)
        sents > 53
        cphr3 <= 1
        abig4 > 4
        sentcount <= 10
        ->  class verybad  [0.631]
```

Figure 3. Rules Discovered by C5.0

parameter sets rated 'verygood' and the rest all had a 'good' set.

C5.0 requires a discretized classification value, the normalized KPIA rating in this case. KPIA ratings ranging from 0.0 to 3.5 were therefore mapped to the five values [verybad, bad, medium, good, verygood]. We performed discretization using the k-means clustering technique (McQueen 1967) which discovers groupings in a set of data points, assigning the same discrete value to each element within a grouping but different values to elements in different groupings.

The large number of continuous[2] attribute variables and the inherent difficulty of the problem led C5.0 to produce over 6300 rules relating document features and parameter settings to summary quality as measured by KPIA rating. Such a large number of rules also militated for the use of decision trees.

Individual rules were not inspected manually in course of preparing our submission to the Document Understanding Conference. It is, however, central to our work to discover from the rules the document features and summarizer parameters whose values characterize and produce well-rated summaries. A presentation and discussion of the most significant rules in the classification tree is therefore warranted.

6 Identifying Good and Bad Attributes

Figure 3 shows examples of classification rules taken from the decision tree. It presents on its left and right sides respectively those that classify the greatest number of cases into the two values at the limits of the rating scale, 'verygood' and 'verybad'.

Although C5.0 rules use the familiar antecedent → consequent format, qualifiers requiring explanation also appear. The slashed *n/m* pair in parentheses following each rule label indicates how many training examples the rule covered and how many of these fell in the class predicted by the rule. *Lift* is the rule's estimated accuracy[3] divided by the relative frequency of the predicted class. The square-bracketed value

[2] C5.0 tends to produce many rules from continuous data because every distinct numeric value can figure in a rule.

[3] $(n-m+1)/(n+2)$. The training set contained 144,806 cases.

near the outcome of each rule tells us the confidence with which the rule's prediction is made.

Negative rules (showing attributes and values to avoid) can be as important as positive ones (showing attributes and values to prefer). Rule 1, covering 1/3 of all cases and predicting 1/6 of all outcomes, is by far the most important rule in the set and it is negative. Rule 1 says very simply that the NPSeeker keyphrase extractor does not produce good results. Rule 4 is the most important positive rule. It suggests that very good summaries of six sentences or more (*sentcount*) result when the Kea keyphraser (*keyphrase*) extracts more than three keyphrases (*sentcount*) from short documents (*sents* less than 53 sentences).

It is interesting that these two rules involve only a single document feature, despite document features composing 20 of the 26 attributes in each summary description. That situation changes with higher numbered rules. Positive Rule 6 says that the Extractor keyphraser (*keyphrase*) gives good results when there are 3-instance bigrams (*abig3*) and few content phrases (*cphr*). Since a keyphraser is likely to select just such syntactic elements as bigrams, the two conditions make sense together. So do those of the next negative rule, Rule 11, which states that documents with fewer than 748 sentences but more than 248 paragraphs do not summarize well. Since such documents have quite short paragraphs (~ 3 sentences), this also makes sense.

From this point onward however it becomes difficult to draw simple conclusions from the attributes and values that figure in rules. This does not suggest the rules are ill-founded. They may in fact efficiently partition the training data space, but they and the preponderance of the 6000 rules remaining do so in a more complicated way than people can easily follow.

Two summarizer parameters do not figure in any important rule, negative or positive: *segmenter* (the segmentation technique used), and *matchtype* (match exactly or with stemmed keyphrases). If confirmed, this finding can lead to two different conclusions, one less and one more likely: (a) good summaries might be produced without involving two of the three stages currently used in summary generation; (b) the *choice* between the available segmenters and matching methods does not affect the quality of summaries, so that the faster segmenter and the cheaper matching can be used.

7 Conclusions and Future Work

The 'average coverage' measure employed by NIST in DUC-2001 scored our summaries of single documents third best, while summaries of multiple documents were ranked towards the end of the set of twelve submissions. The first result may be accounted for in part by the fact that our system adjusts operations to the document submitted. Insofar as the various 'black box' segmenters, keyphrasers etc are better suited to one document than another and this information communicates through the machine learning loop, the TS program has the capacity to pick the 'black box' most suited to a document.

Our poorer results in summarizing multiple documents might be explained objectively. The strategy of favouring locality in sentence selection presumes that the author of the source document has followed an orderly plan in presenting his message; the strategy attempts to take advantage of this orderliness. Multiple documents are very unlikely to have an equally coherent structure. It is a fact which the research community recognizes by having a separate category for multiple document summaries. In a system based on sentence extraction preferring locality, especially in a short summary, almost guarantees that the summary sentences will be chosen from just one or two documents. Information contributed by other documents—and there almost always is some—will be ignored. In such a case picking the globally highest-ranked sentences may be a better practice. On the other hand, such a summary runs a greater risk of very low coherence, because almost every summary sentence comes from a different document.

This paper has presented a framework for text summarization based on the generate-and-test model. We generate a large set of summaries by enumerating all plausible values of six parameters, and assess their quality by measuring them against the document abstract. Ranges of parameters values are identified by supervised machine learning which could be used with some confidence to summarize new documents lacking abstracts. We speculate that the summarizer's reliability would increase as the training corpus domain and that of new documents became more similar.

At least four issues should be considered in the near future.

- **Feedback loop:** Manual operation meant in this experiment that parameters were assessed only once. Useful parameter values should be kept, new parameter values added, and the process repeated until no improvement results.
- **Improve summary description and rating:** We currently ignore document characteristics when summarizing. It may be that different documents require different methodologies—long versus short texts, or dense versus colloquial ones.Description of the document should be included in the assessment process. Alternative measures of summary quality are also desirable.
- **Full automation:** At present parameter assessment is linked to summary generation manually. This makes runs time-consuming and impedes implementing the feedback loop mentioned earlier. An important goal is to automate the system fully so it can run on its own.
- **Integrate other techniques/ methodologies into the Summary Generator:** Our current system only experimented with a restricted number of methodologies and techniques. Once we have a fully automated system, nothing will prevent us from trying a variety of new approaches to text summarization that can then be assessed automatically until an optimal one has been devised.

8 Acknowledgements

This work has been supported by the Natural Sciences and Engineering Research Council of Canada and by our University.

9 References

Choi, F. 2000. Advances in domain independent linear text segmentation. In Proceedings of ANLP/NAACL-00.

Hearst, M. 1997. TexTiling: Segmenting text into multi-paragraph subtopic passages. Computational Linguistics 23 (1), pp. 33-64.

Goldstein, J., M. Kantrowitz, V. Mittal, and J. Carbonell. 1999. Summarizing Text Documents: Sentence Selection and Evaluation Metrics. In Proceedings of ACM/SIGIR-99, pp.121-128.

Kan, M.-Y., J. Klavans and K. McKeown. 1998. Linear Segmentation and Segment Significance. In Proceedings of WVLC-6, pp. 197-205.

Klavans, J., K. McKeown, M.-Y. Kan and S. Lee. 1998. Resources for Evaluation of Summarization Techniques. In Proceedings of the 1st International Conference on Language Resources and Evaluation.

MacQueen, J. 1967. Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical statistics and probability, (1), pp. 281-297.

Mittal, V., M. Kantrowitz, J. Goldstein & J. Carbonell. 1999. Selecting Text Spans for Document Summaries: Heuristics and Metrics. In Proceedings of AAAI-99, pp. 467-473.

Quinlan, J.R. 1993. C4.5: Programs for Machine Learning. Morgan Kaufman, San Mateo, CA.

Turney, P. 2000. Learning algorithms for keyphrase extraction. Information Retrieval, 2 (4), pp. 303-336.

Witten, I.H., G. Paynter, E. Frank, C. Gutwin and C. Nevill-Manning. 1999. KEA: Practical automatic keyphrase extraction. In Proceedings of DL-99, pp. 254-256.