Learning How Best to Summarize

Terry COPECK, **Stan SZPAKOWICZ, Nathalie JAPKOWICZ**
School of Information Technology & Engineering
University of Ottawa
800 King Edward, P.O. Box 450 Stn. A
Ottawa, Ontario, Canada K1N 6N5
{terry, szpak, nat}@site.uottawa.ca

Abstract

We present a summarizer that uses alternative modules to perform its key tasks. Such a *configurable* summarizer benefits from machine learning techniques that help discover which configuration performs best for documents in general or for those characterized by particular lexical features. We determine the quality of an automatically generated summary by computing a measure of its coverage of the content phrases in a model summary known to be of good quality. Machine learning demonstrates significant improvement in this measure over the average value of summaries produced from all configuration parameter settings.

We perform text summarization by *extracting* the most representative sentences from a *single* document. Such summaries tend to be *informative*, rather than indicative; and *generic*, rather than biased towards the interests or purposes of any reader. This is how we situate our work in the gradually emerging classification of summary types which was recently recapitulated in Kan, Klavans and McKeown (2002).

1 System Design

Our work on summarization began several years ago in the context of a larger project with multiple objectives, in which summary generation would be one of the tools. We therefore came to this task for practical reasons and with few hypotheses in play. From the outset it was clear that the approach we envisioned could be implemented in large part by combining existing NLP research software which were either in the public domain or freely available from their authors. It suited our purposes not to reinvent any more wheels than really necessary. It also seemed a good idea to use software which is known to the NLP research community and has already been subject to its scrutiny. We therefore conceived of our summarization system as a testbed in which we could evaluate how well alternative *black boxes* developed by others cooperate as they perform one of the key tasks involved in our method of summarization. The rationale inherent in this architecture has itself in due course provided a research agenda.

We assumed that a summary composed of neighboring, thematically-related sentences should read better, and perhaps also convey more information, than one in which each sentence is independently chosen from the document as a whole and thus likely bears no relation to its neighbours. This assumption led to a process in which the text to be summarized is first broken into *segments*, which are sequences of adjacent sentences talking about the same topic. In a separate operation *keyphrases* are extracted from the whole text and ranked. Each sentence in the text is then rated according to the number of highly-ranked keyphrases which it contains, and each segment is ranked according to the ratings of the sentences which it contains. The summary is then constructed by picking sentences from those that exceed a threshold count of keyphrases in the most highly ranked segments, moving from the best segment to the next best until the required number of sentences have been accumulated. These sentences are put in document order and presented to the user as a summary of the document.

2 Processing

Because our summarizer was at first designed to be a functional component of a larger project, it was necessary to handle real world texts which, although coded in plain ASCII, may have a variety of extraneous formatting: pagination, embedded figures and tables and, most crucially, columnar format. Although it took some effort to install the black boxes in wrappers that brought them to a common interface, most of our attention was spent on translating input text to the "NLP normal" form — one sentence per line, one extra line between paragraphs — required by most of the black boxes. This form is also made necessary by the task itself; sentence extraction presumes a text in which sentences are identified correctly. Recognizing sentence and paragraph boundaries and extraneous material such as tables interpolated into the flow of text are legitimate research issues in themselves. They are, however, not inherently

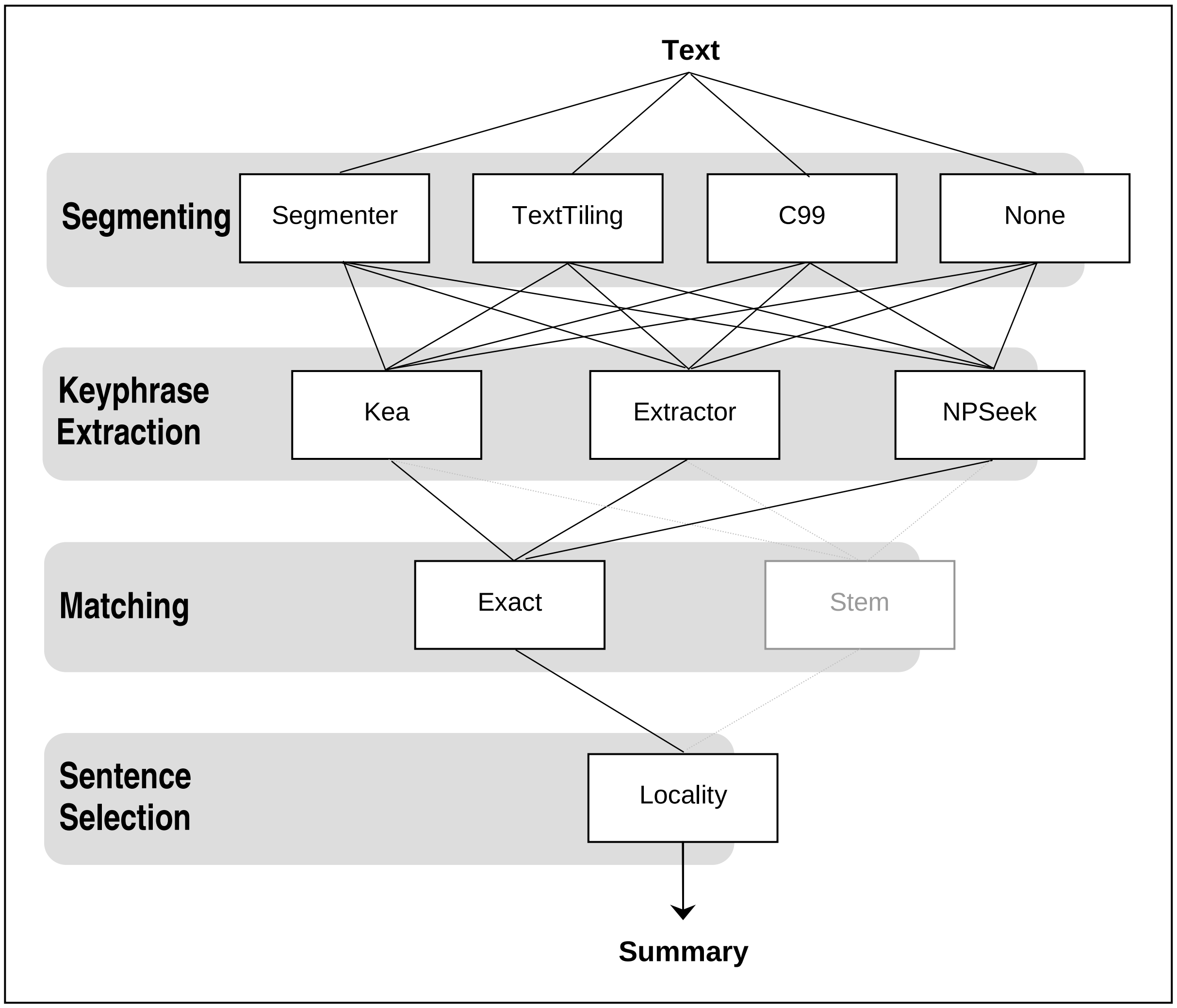

Figure 1. Summarizer Program Architecture

part of summarization, so we will not discuss them further here.

In the course of processing of input text prior to summarization we create a *data model* of the document which records its various lexical features on a per-sentence basis as well as details of sentence and paragraph order. This record is the basis for the document characterization which plays a role in summary evaluation discussed in Section 4.

The key tasks for which we used alternative external programs are document segmentation, keyphrase extraction, and phrase matching. The Columbia University Segmenter (Kan, Klavans and McKeown 1998), Hearst's TextTiling program (1997) and the C99 program (Choi 2000) were used to divide a document into segments. A fourth option implemented recently turns off segmentation and treats the document as a single segment. A moment's thought may show that this is the control case for our hypothesis about the benefit of locality in sentence selection; picking the highest-rated sentences from the entire text defeats locality. To extract keyphrases we used Waikato University's Kea (Witten, Paynter, Frank, Gutwin and Nevill-Manning 1999), the Extractor program from the Canadian NRC's Institute for Information Technology (Turney 2000), and NPSeek, a program developed at the University of Ottawa (Barker and Cornacchia 2000). The Kea program suite also provided modules to match phrases exactly and in stemmed form. Figure 1 shows the architecture of the summarization program constructed around these black boxes (stemmed matching is dimmed deliberately).

3 Operational Parameters

To run, our program must be told which black box to use for each of the three key tasks where we have choice. Three other aspects of the process or the task have been judged important enough to require parametrization: 1) the size of the summary in sentences, words, or as a proportion of the document; 2) the number of keyphrases to use in rating sentences and segments; 3) the threshold number of keyphrase matches needed for a sentence to qualify as a candidate for a summary. There are 24 unique specifications for black boxes alone; each of the three additional parameters takes a count rather than an enumerated value, so it is quite feasible to produce hundreds if not thousands of summaries for a single document.

This is computationally expensive. Results from DUC 2001 led us to shrink this space by dispensing with stemmed matching (which for this reason is shown dimmed in Figure 1), fixing the keyphrase threshold at two and limiting the range of the other two count parameters, summary size and number of keyphrases.

4 Evaluation

How good are any of these summaries? Answering that question has proven to be quite difficult (Goldstein, Kantrowitz, Mittal and Carbonell 1999), and it is currently the object of quite intense research interest, of which the two Document Understanding Conferences (DUC) sponsored by NIST are notable examples. Summary evaluation has become a research goal in itself.

A variety of statistical measures of summary content — kappa, relative utility, and the traditional IR metrics of precision and recall — are computed in systems like the MEAD Eval toolkit (Radev, Teufel, Lam and Saggion 2002) or SEE (Lin 2001) that automate or assist the assessment of summaries against the original texts on which they are based. We have, however, chosen to compare our summaries against the *gold standard* of ones written by human authors. Such summaries are not easily come by. We had therefore initially settled on academic papers[1] as a subject genre and have adopted the authors' abstracts which begin such papers as a practical approximation to the gold standard summary we seek, an approach proposed by Mittal, Kantrowitz, Goldstein and Carbonell (1999). Who better than the author knows the intended content of a paper?

[1] Initially we used a one-million-word corpus of 75 academic papers published between 1993-1996 in the online *Journal of Artificial Intelligence Research*.

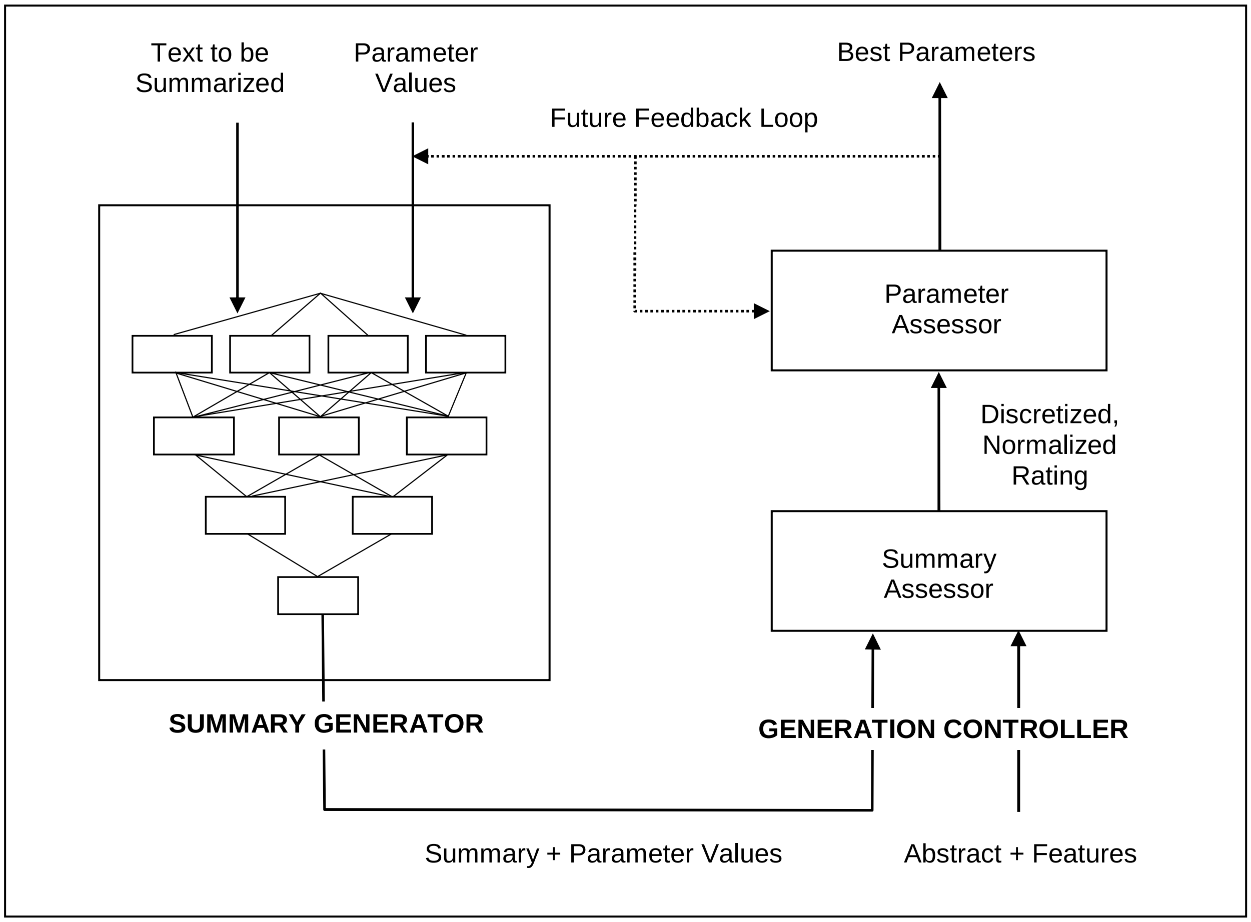

Figure 2. Text Summarization Learner Architecture

We use a variant of the keyphrase extraction technique used in the summarization program to evaluate a summary against the reference summary. That gold standard is broken down into a set of unique content keyphrases by processing it against a stoplist of 980 closed-class and common words. The resulting *keyphrases in abstract* (KPiAs) differ from those used to produce the summary under evaluation in that they compose a comprehensive description of the abstract rather than being limited to a small number of the most highly ranked ones.

A summary is ranked according to the degree to which it covers the set of KPiAs. Its score is computed using the following formula:

$$\sum 1.0 * \text{KPiA}_{\text{unique}} + 0.5 * \text{KPiA}_{\text{duplicate}}$$

The formula computes a total for the sentence, assigning a weight of 1.0 to each KPiA which is added to the coverage set and 0.5 to each that repeats a KPiA which is already in the set.

5 Learning

The combination of a wholly automated evaluation technique and a large number of instances to which to apply this technique is a textbook application for machine learning. Accordingly, we embedded the summarization program in a framework which automates its operation and the evaluation of its results and then applies the C5.0 decision tree and rule classifier (Quinlan 2002) to infer rules about which parameter settings produce highly-rated or poorly-rated summaries. This architecture appears in Figure 2. To permit comparisons between documents, each summary KPiA rating is normalized by dividing it by the total count of unique KPiAs found in the same number of sentences ranked highest in the

FEATURE	DESCRIPTION	FEATURE	DESCRIPTION
chars	number of characters in the document	cphr3	number of 3-instance content phrases
words	number of words in the document	cphr4	number of 4-or-more instance content phrases
sents	number of sentences in the document	cbig	number of 1-instance content bigrams
paras	number of paragraphs in the document	cbig2	number of 2-instance content bigrams
kpiacnt	number of kpia instances	cbig3	number of 3-instance content bigrams
conncnt	number of (Marcu) connectives	cbig4	number of 4-or-more instance content bigrams
pncnt	number of PNs, acronyms	abig	number of 1-instance bigrams
contcnt	number of content phrases	abig2	number of 2-instance bigrams
cphr	number of 1-instance content phrases	abig3	number of 3-instance bigrams
cphr2	number of 2-instance content phrases	abig4	number of 4-or-more instance bigrams

Table 1. Document Features

document on KPiA count. This establishes the highest possible rating for a summary of a given size. The set of ratings for summaries of all documents is then discretized into the five values[2] *verybad*, *bad*, *medium*, *good*, *verygood* using MacQueen's (1967) k-means clustering technique. C5.0 then induces rules based on these ratings and parameter values. Much of the learner's operation has been automated, most recently the identification of best parameters in the C5.0 output and the application of k-means clustering, and the job of completely automating operation of the system is under way.

Although our first experiments involved a data vector containing only the settings of the six parameters that regulate the production of summaries, availability of the document data model soon led us to add the features listed in Table 1 in order to characterize the document being summarized. These features are all counts: the number of characters, words, sentences and paragraphs in the document; and the number of other syntactic elements, such as KPiAs, connectives and proper nouns. We also count content phrases (substrings between stoplist entries), bigrams (word pairs) and content bigrams. We record how many of these latter three primitive lexical items occur once, twice, three times, or more than three times in the document.

[2] C5.0 requires a discrete outcome variable. The use of five values was determined heuristically.

Adding features has the effect of asking C5.0 to discover rules identifying which parameter values produce good summaries for documents with certain features, rather than for documents in general. Further active investigation over the last year has not yet identified additional features which might be used effectively to characterize documents.

Our experiments clearly show the benefit of machine learning. Choosing parameter values randomly produces an average rating of 2.79, the *medium* value that is to be expected from a population of values that have been normalized. When the best parameter settings for each text are chosen, however, the average rating increases two full grades to 4.78 or *verygood*.

6 Document Understanding Conference

We participated in the Document Understanding Conferences in 2001 (Copeck, Japkowicz and Szpakowicz 2002) and 2002. In DUC, NIST first provides participants with a corpus of training documents together with abstracts written by skilled editors, and then a set of similar test documents which each participant is to summarize in a manner consistent with that inferred from the training set. Our participation involved slight modification to our learning system. Rather than constructing rules that relate features and parameters to KPiA ratings, C5.0 used the training data to build a classification tree recording these

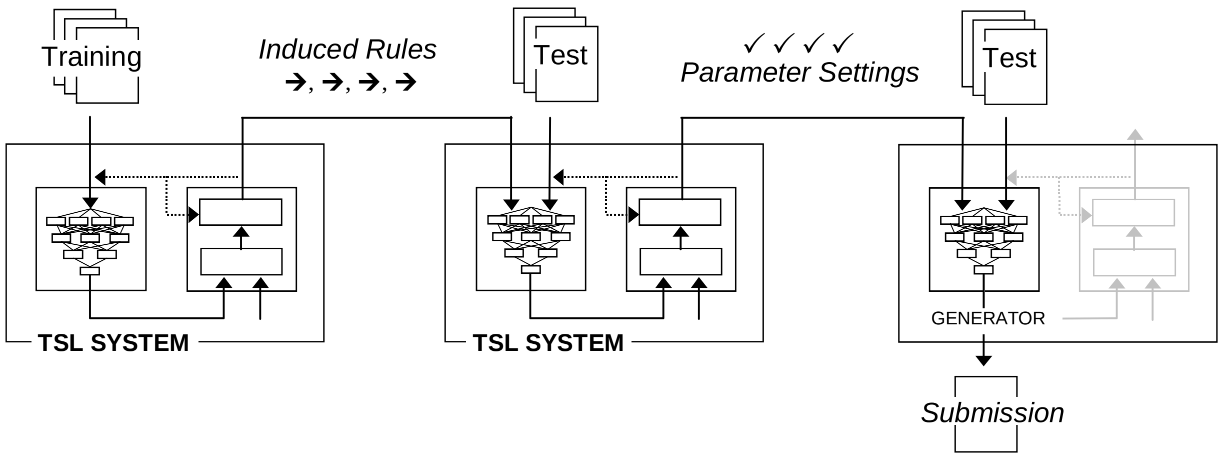

Figure 3. Methodology for DUC 2002

same relationships. The test data, which is composed of document features only, was then classified against this structure and the parameters that gave the best summary for the training document most similar to each test document were used to produce the summaries submitted to the conference. Figure 3 depicts the process.

Ten-fold cross validation on the training data shows 16.3% classification error (std. dev. .01). The 5x5 confusion matrix further indicates that 80% of misclassifications were to the adjacent value in the scale. This suggests that the rules are reasonably trustworthy. The test run showed that they are productive, and that the ratings they seek to maximize are well-founded.

For DUC 2002 we added adaptive boosting (Freund and Schapire 1996) to the learning process. Adaptive boosting, or ADABOOST, improves the classification process by iteratively generating a number of classifiers from the data, each optimized to classify correctly the cases most obviously misclassified on the previous pass. The votes of a population of these classifiers generally provide better outcomes than do the decisions of any single classifier.

NIST provides participants with a full suite of submissions and analysis material after each event as an invitation to get involved in judging results. We used the human summaries of the 2001 test documents to augment training data for 2002 from 308 to 590 documents. Our own evaluation of our results for 2001 varied from that computed by Paul Over (2001) for no obvious reason, so we held back from reporting results for DUC 2002 at the July workshop on automatic summarization occurring in conjunction with ACL-02. In 2001 we did well on single document extractive summaries and worse than average on multi-document summaries. Because our technique is not particularly suited to summarize more than one document at a time, and because we are not entirely comfortable with the notion of incorporating material from more than one document in a single summary, we chose to participate only in the single document track in 2002.

In DUC 2002 our submission ranked somewhat below the middle of the pack of 13 contributors in the single document track. This is lower than our performance the previous year, though ranking alone is not a clear indication of relative decrease in performance.

A measure used both in the training phase and the test phase offers insights into the state of the overall process. We compute the discretized KPiA rating for all summaries of documents in the training set. We also infer from the classification tree a KPiA rating for each summary of every document in the test set. Distribution of the five rating values across any set of summaries establishes the limits of the range of possible outcomes. A comparison of KPiA distributions for DUC 2001 with their counterparts for DUC 2002 appears in Table 2. Rows show the percentage of

	2 0 0 1				2 0 0 2			
RATING	TRAIN	TEST		SUB	TRAIN	TEST		SUB
verygood	11%	14%	+ 2.00	79%	13%	15%	+ 2.68	77%
good	12%	21%		21%	18%	1%		2%
medium	20%	19%			24%			3%
bad	18%	21%			26%	4%		12%
verybad	39%	25%			19%	80%		7%
AVERAGE	***2.35***	***2.79***		***4.79***	***2.80***	***1.68***		***4.31***

Table 2. KPiA Summary Rating Distributions in DUC Corpora

summaries rated, or inferred to be, in the indicated class. Each column concludes with the average rating for all its summaries. In 2001 test and training data were similar, and machine learning, indicated by the gray arrow, was able to find settings for each test document that produced at least a good summary.

Contrast this with the situation in 2002. The training data ratings are akin to those for the 2001 data, but ratings of the test data are widely at variance with those of the three other document collections. Fully 80% of the summaries of the test data collection are inferred to be *verybad*. Despite machine learning outperforming its counterpart in 2001, there were simply no well-rated summaries that could be produced for approximately 22% of the 2002 test documents. This situation almost guaranteed the low evaluation produced by the NIST judges.

The most obvious explanation for these circumstances would be an inconsistency between the 2002 training and test data. Our system would exhibit this in the values it computes for the 19 features used to model each document for machine learning. However, despite the presence of outliers in each set of documents—texts much longer or shorter than the average—inspection shows that the distribution of each feature in one collection resembles its counterpart in the other to a far greater degree than do the two problematic KPiA rating distributions. In addition, where individual features of a document are related to one another, distributions in the two collections also differ in consistent ways. In sum, the test and training collections appear to resemble one another to the degree one would expect.

We have therefore ruled out the most obvious explanation for our results. Although we have not yet been able to find a reason why our 2002 training summaries had such low ratings, we continue to investigate.

7 Conclusions and Future Work

We intend in future:

- to complete automating the operation of the entire text summarization learning system;
- to continue to look for lexical features which describe documents effectively. In this regard commercial keyphrasers identify entities as place, city, state and country; company, organization and group; day and year; percent and measure; person; and property. Some of these may merit recording;
- to investigate the SEE and MEADEval summarization evaluators;
- to merge the output sets of two or all three keyphrasers, providing new alternatives. To what degree do the keyphrasers agree in their selections? How would merging sets of keyphrases be accomplished? What effect would use of a merged set have on summary output?

- to consider new techniques for sentence selection, which may involve factors other than keyphrase matching. For instance words marking anaphora may make a sentence less well suited to a summary;
- to continue looking to add new segmenters and keyphrasers to our system.
- to investigate further the role of segmentation and locality in text summarization.

Acknowledgements

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

References

Barker, K. and N. Cornacchia. 2000. Using Noun Phrase Heads to Extract Document Keyphrases. Proc 13th Conf of the CSCSI, AI 2000 (LNAI 1822). Montreal, 40-52.

Choi, F. 2000. Advances in domain independent linear text segmentation. Proc ANLP/NAACL-00, 26-33.

Copeck, T., N. Japkowicz and S. Szpakowicz. 2002. Text Summarization as Controlled Search. Proc 15th Conf of the CSCSI, AI 2002 (LNAI 2338), Calgary, 268-280.

Freund, Y., Shapire, R. 1996. Experiments with a new boosting algorithm. Proc 13th ICML. Morgan Kaufmann. 325-332.

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

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

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

Kan, M.-Y., J. Klavans and K. McKeown. 2002. Using the Annotated Bibliography as a Resource for Indicative Summarization. Proc LREC 2002, Las Palmas, Spain. 1746-1752.

Lin, C.-Y. 2001. SEE—Summary Evaluation Environment. www.isi.edu/~cyl/SEE/.

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

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

Over, P. 2001. Introduction to DUC 2001: an Intrinsic Evaluation of Generic News Text Summarization Systems. www-nlpir.nist.gov/projects/duc/duc2001/pauls_slides/duc2001.ppt.

Quinlan, J.R. 2002. C5.0: An Informal Tutorial. www.rulequest.com/see5-unix.html.

Radev, D., S. Teufel, W. Lam and H. Saggion. 2002. Automatic Summarization of Multiple Documents (MEAD) Project. www.clsp.jhu.edu/ws2001/groups/asmd/.

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

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