Using Gene Expression Programming to Construct Sentence Ranking Functions for Text Summarization

Zhuli Xie, Xin Li, Barbara Di Eugenio, Peter C. Nelson
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607, U.S.A.
zxie@cs.uic.edu, xli1@cs.uic.edu,
bdieugen@cs.uic.edu, nelson@cs.uic.edu

Weimin Xiao, Thomas M. Tirpak
Physical Realization Research Center of
Motorola Labs
Schaumburg, IL 60196, U.S.A.
awx003@motorola.com,
T.Tirpak@motorola.com

Abstract

In this paper, we consider the automatic text summarization as a challenging task of machine learning. We proposed a novel summarization system architecture which employs Gene Expression Programming technique as its learning mechanism. The preliminary experimental results have shown that our prototype system outperforms the baseline systems.

1 Introduction

Automatic text summarization has been studied for decades (Edmundson 1969) and is still a very active area (Salton et al. 1994; Kupiec et al. 1995; Brandow et al. 1995; Lin 1999; Aone et al. 1999; Sekine and Nobata 2001; Mani 2001; McKeown et al. 2001; Radev et al. 2003). Only a few have tried using machine learning to accomplish this difficult task (Lin 1999; Aone et al. 1999; Neto et al. 2002). Most research falls into combining statistical methods with linguistic analysis. We regard the summarization as a problem of empowering a machine to learn from human-summarized text documents. We employ an evolutionary algorithm, Gene Expression Programming (GEP) (Ferreira 2001), as the learning mechanism in our Adaptive Text Summarization (ATS) system to learn sentence ranking functions. Even though our system generates extractive summaries, the sentence ranking function in use differentiates ours from that of (Edmundson 1969; Sekine and Nobata. 1999; Goldstein et al. 1999) who specified it to be a linear function of sentence features. We used GEP to generate a sentence ranking function from the training data and applied it to the test data, which also differs from (Lin 1999) who used decision tree, (Aone et al. 1999; Kupiec et al. 1995) who used Bayes's rule, and (Neto et al. 2002) who implemented both Naïve Bayes and decision tree C4.5.

This paper presents our approach, details the system architecture, and discusses preliminary experimental results. Conclusions and future work are outlined at the end.

2 Background

2.1 Gene Expression Programming

Gene Expression Programming (GEP), first introduced by (Ferreira 2001), is an evolutionary algorithm that evolves computer programs and predicts mathematical models from experimental data. The algorithm is similar to Genetic Programming (GP), but uses fixed-length character strings (called *chromosomes*) to represent computer programs which are afterwards expressed as expression trees (ETs). GEP begins with a random population of candidate solutions in the form of chromosomes. The chromosomes are then mapped into ETs, evaluated based on a fitness function and selected by fitness to reproduce with modification via genetic operations. The new generation of solutions goes through the same process until the stop condition is satisfied. The fittest individual serves as the final solution. GEP has been used to solve symbolic regression, sequence induction, and classification problems efficiently (Ferreira 2002; Zhou 2003). We utilized GEP to find the explicit form of sentence ranking functions for the automatic text summarization.

2.2 Sentence Features

In our current system, every sentence ***s*** is represented by five normalized features:

- **Location of the Paragraph (P):**

$$P = Y / M \quad (1)$$

where ***M*** is the total number of paragraphs in a document; ***Y*** is the index of the paragraph ***s*** belongs to.

- **Location of the Sentence (S):**

$$S = X / N \quad (2)$$

where ***N*** is the total number of sentences in the paragraph; ***X*** is the index of sentence s.

- **Length of the Sentence (L):**

The length of the sentence is the number of words it contained, i.e., ***l(s)***, normalized by Sigmoid function:

$$L = \frac{1-e^{-\alpha}}{1+e^{-\alpha}}, \alpha = \frac{l(s)-\mu(l(s))}{std(l(s))} \quad (3)$$

Where ***u(l(s))*** is the average length of sentences, and ***std(l(s))*** is the standard deviation of the sentence lengths.

- **Heading Sentence (H):**

H = 1, if ***s*** is a title, subtitle or heading, 0 otherwise.

- **Content-word Frequencies (F):**

$$F(s) = \frac{1-e^{-\alpha}}{1+e^{-\alpha}},\ \alpha = \frac{CW(s)-\mu(CW(s))}{std(CW(s))} \quad (4)$$

$$CW(s) = -\sum_{i=1}^{k} \log[Freq(w_i)], w_i \in s. \quad (5)$$

where ***Freq(w_i)*** is the frequency of ***w_i*** in that document; ***μ(CW(S))*** is the mean of all the sentence scores, and ***std(CW(s))*** is the standard deviation.

2.3 Sentence ranking function

We assume that for a certain type of documents, the mechanism to perform summarization would be the same. Therefore, we only need to find one algorithm that links a collection of documents and their corresponding summaries. We process the text summarization learning task in two stages: training and testing. In the training stage, a set of training documents with their summaries are provided, and the text features are preprocessed using statistical methods and natural language processing methods as defined in 2.2, then each sentence in a document is scored based on a sentence ranking function constructed by GEP. Fitness value of the summarization task is the similarity between the summary produced by the machine and the summarization text of training document. The top ***n*** ranked sentences[1] will be returned as the summary of that document and presented in their nature order. In the testing stage, a different document set is supplied to test the similarity between the machine summarized text and the human or other system summarized text.

3 System Architecture

In addition to the traditional way of extracting the highest ranked sentences in a document to compose a summary as in (Edmundson 1969; Lin 1999; Kupiec et al. 1995; Brandow 1995; Zechner 1996), we embedded a machine learning mechanism in our system. The system architecture is shown in Figure 1 where the GEP module is highlighted. In the training stage, each of the training documents is passed to the GEP module after being preprocessed into a set of sentence feature vectors. The GEP runs ***m*** generations, and in each generation a population of ***p*** sentence scoring functions in the form of chromosomes in GEP is generated. Every candidate scoring function is then applied to sentence feature vectors from every training document and produces a score accordingly. Then all sentences in the same training document are ranked according to their scores, and ***n*** sentences with top scores are selected as an extract. The next step is to measure how similar the extract is to the objective summary. As discussed by (McLellan et al. 2001; Goldstein et al. 1999; McKeown et al. 2001), evaluating the quality of a summary often requires involvement of human subjects. This is almost impractical in a machine learning procedure. Thus we chose an alternative similarity measure as the approximation, i.e. a cosine function that is often seen in Information Retrieval to calculate the relevance of two documents, to compute the similarity between an extract and the objective summary. We compute the similarity values for each of the obtained extracts and their objective summaries respectively, and feed the results into the Fitness Calculation module to get a fitness measure for the current candidate sentence ranking function under consideration:

$$Fitness = Avg(Similarity(E_i, O_i)), \quad (6)$$

where ***E_i*** is the extract of the ***i***-th document in the training set and ***O_i*** is its objective summary.

After the fitness value for every chromosome in the current generation is computed, the GEP population undergoes all genetic operators to produce the next generation. After the specified number of generations has been reached, the final best chromo-

[1] The number of sentences extracted by the GEP module can be a variable, which is decided by the required number of words in a summary. Or it can be a specified percentage of the total number of sentences in the document.

some is returned as an optimal sentence ranking function for the training set and is ready to use in a test document to produce an extractive summary.

4 Experiments

We randomly selected 60 documents from the CMPLG corpus[2] for our experiments. The only restriction is that each document has an abstract provided which will serve as the objective summary. Among these 60 documents, 50 are used for training and the remaining 10 are used for testing. The function set for the GEP to evolve sentence ranking functions includes (+, -, *, /, power, sqrt, exp, log, min, max, and constant 1, 2, 3, 5, 7). The length of the chromosome is 128. Other GEP control parameters are set as follows: population, 256; probability of crossover, 0.5; probability of mutation, 0.2; probability of rotation, 0.2; generations, 10,000-50,000 (in five runs). Our system has produced a five-sentence extractive summary for each of the testing documents, and calculated the similarity between the produced summary and the abstract coming along with the document.

Ideally, we would like to compare our system with other summarizers. However, due to the unavailability of other summarization systems to perform the same task, we designed three baseline methods, namely lead-based, randomly-selected, and random-lead-based, to generate summaries for performance comparison, which were also adopted by (Brandow et al. 1995; Zechner 1996; Radev et al. 2003). The baseline methods are detailed as follows:

- The lead-based method selects the first sentences from the first five paragraphs as the summary of each of the testing documents.
- The randomly-selected method chooses five sentences from a document at random to compose a summary.
- The random-lead-based method chooses five sentences among the first sentences from all paragraphs in the document at random.

We performed the random selection 1,000 times, and calculated the average similarity of the testing documents for each of the random-based methods. The experimental results are plotted in Figure 2, which have demonstrated that our system outperforms all three baseline methods.

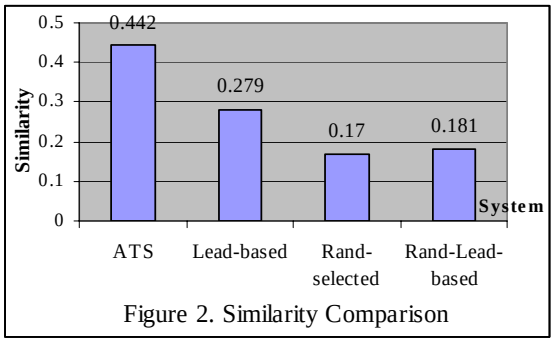

Figure 2. Similarity Comparison

One sample sentence scoring function learned by the GEP is as follows:

$$score(s) = \frac{7 - S}{F(\sqrt{P} + 3)}, \qquad (7).$$

5 Conclusions and Future Work

In this paper, we have presented a prototype summarization system which employs GEP as its learning mechanism for sentence ranking function. In the preliminary experiments for performance testing, our system outperforms the baseline methods by 58%-160% when generating summaries for 10 documents. However, the value of the average similarity gained by our system is not as high as we would like. The reason most likely lies in the fact that the styles of the objective summaries written by humans vary a lot or even conflict with each other. In other words, they do not possess many common features that are a must for high value of similarity between two texts. Using content-words and the cosine function to measure the similarity may not be an ideal evaluation metric, neither is it an ideal fitness measure in the GEP learning mechanism. Our future

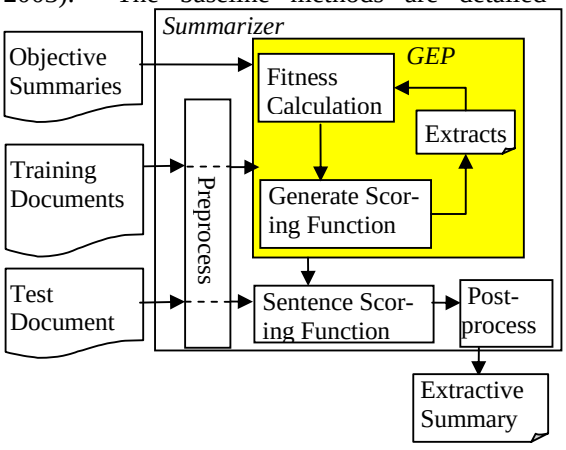

Figure 1: System Architecture

[2] CMPLG corpus is composed of 183 documents from the Computation and Language (cmp-lg) collection, which has been marked up in XML. The documents are scientific papers which appeared in association for Computational Linguistics (ACL) sponsored conferences.

research will further study what kinds of similarity measure can be obtained from raw texts without involvement of human subjects. Moreover, we plan to cluster collected documents to make every cluster contains articles summarized in a similar style. We will also explore other sentence features, such as sentence cohesion, semantic meaning, and rhetorical relations, for an ideal uniform sentence ranking function.

6 Acknowledgements

Our thanks go to the Physical Realization Research Center of Motorola Labs for their support of this research project.

References

Aone, C., Gorlinsky, J., Larsen, B., and Okurowski, M. E. 1999. A Trainable Summarizer with Knowledge Acquried from Robust NLP Techniques, *Advances in Automatic Text Summarization*, pages 71-80. The MIT Press, Cambridge, Massachusetts.

Brandow, R., Mitze, K., and Rau, L. F. 1995. Automatic condensation of electronic publications by sentence selection. *Information Processing and Management*, 31(5):675-685.

Edmundson, H. 1969. New methods in automatic abstracting. *Journal of ACM*, 16(2):264-285.

Ferreira, C. 2001. Gene Expression Programming: A New Adaptive Algorithm for solving problems. *Complex Systems*, 13(2):87-129.

Ferreira, C. 2002. *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence*. Angra do Heroismo, Portugal

Goldstein, J., Kantrowitz, M., Mittal, V., and Carbonell, J. 1999. Summarizing Text Documents: Sentence Selection and Evaluation Metrics, in *Proc. SIGIR '99*, pages 121-128. Berkeley, California.

Kupiec, J., Pedersen, J., and Chen, F. 1995. A trainable document summarizer. In *Proc. 18th ACM-SIGIR Conference*, pages 68-73. Seattle, Washington.

Lin, C. 1999. Training a Selection Function for Extraction. In *the 8th International Conference on Information and Knowledge Management* (CIKM 99), Kansa City, Missouri.

Mani, I. 2001. *Automatic Summarization*, John Benjamins Publishing Company, Amsterdam/Philadelphia.

McKeown, K. R., Barzilay, R., Evans, D., Hatzivassiloglou, V., Kan, M. Y., Schiffman, B., Teufel, S. 2001. Columbia Multi-Document Summarization: Approach and Evaluation, in *Proceedings of the Document Understanding Conference (DUC01)*. Edmonton, Canada.

McLellan, P., Tombros, A., Jose, J., Ounis, I., and Whitehead, M. 2001. Evaluating summarisation technologies: A task-oriented approach. In *Proc. 1st International Workshop on New Developments in Digital Libraries (NDDL-2001), International Conference on Enterprise Information Systems (ICEIS 2001)*, pages 99-112. Setubal, Portugal.

Neto, J. L., Freitas, A. A., and Kaestner, C. A. A. 2002. Automatic Text Summarization using a Machine Learning Approach. In *Proc. 16th Brazilian Symp. on Artificial Intelligence (SBIA-2002). Lecture Notes in Artificial Intelligence 2507*, pp205-215. Springer-Verlag.

Radev, D. R., Teufel, S., Saggion, H., Lam, W., Blitzer, J., Qi, H., Celebi, A., Liu, D., and Drabek, E. 2003. Evaluation challenges in large-scale document summarization, in *Proc. 41st Annual Meeting of the Association for Computational Linguistics*, pages 375-382. Sapporo, Japan.

Salton, G., Allan, J., Buckley, C., and Singhal, A. 1994. Automatic Analysis, Theme Generation, and Summarization of Machine-Readable Texts. *Science*, 264(3):1421-1426.

Sekine, S. and Nobata, C. 2001. Sentence Extraction with Information Extraction technique. In *Proc. of ACM SIGIR'01 Workshop on Text Summarization*. New Orleans.

Zechner, K. 1996. Fast generation of abstracts from general domain text corpora by extracting relevant sentences. In *Proc. COLING-96*, pages 986-989. Copenhagen, Denmark.

Zhou, C., Xiao, W., Tirpak, T. M., and Nelson, P. C. 2003. Evolving Classification Rules with Gene Expression Programming. *IEEE Transactions on Evolutionary Computation*, 7(6):519 – 531.