Adaptive Learning in Machine Summarization

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

Abstract

In this paper, we propose a novel framework for extractive summarization. Our framework allows the summarizer to adapt and improve itself. Experimental results show that our summarizer achieves higher evaluation scores by adapting to the given evaluation metrics.

Introduction

We propose a novel framework for an automatic text summarization system, called ADAMS (Adaptive Machine Summarization). Our goal is to design a summarizer that is able to adapt to any possible changes in its specifications, i.e. adapt to a new domain, a different task (single or multi-document summarization), or a different evaluation metric. Whereas most empirical, state-of-the-art summarizers (Mani & Maybury 1999) are able to adapt to new domains via re-training on the new domain, adapting to changes to different specifications is harder. In this paper, we discuss how ADAMS adapts to different evaluation metrics. *Adapting to an evaluation metric* may sound like putting a positive spin on unfairly biasing the system in order to obtain better results. To the contrary, the issue is that summary evaluation metrics are still under study and no one has become the standard yet. The field is designing better metrics, where 'better' is e.g. defined as better correlated to human judgements (Lin 2004). If new and better metrics are adopted by the field, state-of-the-art summarizers will have to be modified by their designers in order to produce better summaries. Namely, they will be run as they are, and if they don't perform well with respect to the new metric, their designers will have to modify the underlying model. We believe such work can be avoided if we introduce a summarizer which can adapt and improve itself, like ADAMS.

Adaptive Machine Summarization

Our proposed framework consists of a preprocessing module, a post-processing module, a sentence ranking module, a machine learning module, and an evaluation module.

The learning process works as follows:

In the training stage, a set of training documents, and their corresponding objective abstracts are used as input. In the preprocessing module, each training document is decomposed into a set of sentence feature vectors.

Collection V^+ consists of the sets of sentence feature vectors for all training documents. It is passed into the learning module. Within the learning module, there is a function generation module which uses Gene Expression Programming (GEP) (Ferreira 2001; Xie *et al.* 2004) to generate a set F of sentence ranking functions:

$$F = \{f_l | l = 1, \ldots, p\} \tag{1}$$

where f_l is the sentence ranking function at each learning cycle, and p is an input parameter for the number of functions to be generated by this module. In f_l, operands are sentence features and operators can be any arithmetic, logical, selectional, or functional (*exp*, *min*, *max*, *log*, ...) operators. After F is generated, each f_l is applied to V^+ and produces a score for each sentence feature vector. The sentence feature vectors are ranked according to their scores, and the n highest ranked vectors are selected and their corresponding sentences are retrieved from the text as an extract e_{li}. Thus for each sentence ranking function $f_l \in F$, a set E_l of extracts will be generated:

$$E_l = \{e_{li} | i = 1, \ldots, t\} \tag{2}$$

where l corresponds to the l^{th} ranking function and i corresponds to the i^{th} training document. E_l is then passed to the evaluation module. An evaluation score is computed for every extract e_{li} comparing to the objective abstract A_i and the average score for E_l is returned back to the learning module as a fitness measure for f_l. After the fitness value for every $f_l \in F$ is computed, one learning cycle is completed. Then the GEP population evolves using the genetic operators to produce the next generation. After a specified number of generations, the final best sentence ranking function f_o is returned for use on other unseen documents of the same type. The goal of the learning process is to maximize the average evaluation score to produce higher quality summaries.

Experimental data

We used two corpora in our experiments. The first corpus, COM-LG, contains 178 documents which are from the Computation and Language collection. The second corpus was obtained from the Document Understanding Conferences (DUC). We did not participate in DUC 2004, but here we perform Task 2, creating a short ($\leq$ 665 bytes) summary for a given document cluster. We perform this task even if ADAMS was not designed for multidocument summarization, in order to know how our system compares with the participating systems and to assess whether it can be easily adapted to this new task In our experiments with the DUC data, 60 clusters of documents from DUC 2003 were used for training, and 50 clusters from DUC 2004 were used for testing. Each cluster contains about 10 documents.

Evaluation metrics

Lin (2004) developed a summary evaluation system called ROUGE which automatically determines the quality of a summary by comparing it to objective summaries. He introduced ROUGE-N which is an n-gram recall measure between a candidate summary and a set of reference summaries. ROUGE has been used in DUC 2004 to evaluate the summaries provided by the participants against human-written summaries. We use ROUGE-1 score and cosine similarity as summary evaluation metrics to show ADAMS can adapt to them.

Experimental settings and results

For the COM-LG corpus, we have the following settings:

1. We use the first sentences from the first n paragraphs to compose a lead-based summary (our baseline *Ba*).
2. We use cosine similarity as the evaluation metric to obtain a ranking function F_C.
3. We use ROUGE-1 as the evaluation metric to obtain a ranking function F_R.

In the testing stage, the sentence ranking functions learned from the training stage are applied respectively to the unseen documents to each produce a set of summaries. Then, we use the ROUGE evaluation system and cosine similarity to compare those summaries with the abstracts of the unseen documents. For the DUC data, currently we only use ROUGE-1 as the evaluation metric for both training and testing.

The first row in Table 1 shows the average ROUGE-1 scores of the ADAMS-generated summaries of unseen documents, compared with the corresponding abstracts. The second row gives the average cosine similarity values between the summaries and the abstracts. To evaluate the performances of a summarizer, a lead-based method is widely used in the literature on automatic text summarization and many systems are just about the same as or little better than it (Over & Liggett 2002). Our summarizer, however, greatly outperforms the baseline method (see Table 1). We also see that in the first row, F_R is 14% better than F_C since F_R is adapted to ROUGE-1; and vice versa in the second row, F_C is 22% better than F_R. The results show that when a specific evaluation metric is used as the training target to obtain a sentence ranking function *f*, the summaries produced by *f* for the testing documents will get higher score than those produced by other *f'*. In DUC 2004, 35 systems including one baseline participated in Task 2. The performances of the best, the worst, and the baseline systems are shown in

	Ba	F_C	F_R
ROUGE-1	0.23178	0.42617	0.49235
Cosine Similarity	0.28119	0.37525	0.30744

Table 1: Experimental Results for COM-LG Corpus

System	ROUGE-1
ADAMS	0.333
Baseline	0.324
Best	0.382
Worst	0.242

Table 2: Experimental Results for DUC Corpus

Table 2, along with that of ADAMS. At a ROUGE-1 score of 0.333, ADAMS performs better than the baseline, and lies in the middle compared with the participating systems. This result is very promising given that our system was not designed for multidocument summarization.

Conclusions and Future Work

In this paper we propose a new framework for machine summarization. It allows the summarization system to generate better quality summaries by adapting itself to a "better" summary evaluation metric. Our experiments show that ADAMS does possess the ability to adapt to different evaluation metrics. In future work we will investigate new metrics which can be used in an automatic evaluation environment to measure the overall quality (grammar, fluency, prominence, relativeness, etc.) of the machine generated summaries.

References

Ferreira, C. 2001. Gene expression programming: A new adaptive algorithm for solving problems. *Complex Systems* 13(2):87–129.

Lin, C.-Y. 2004. Rouge: a package for automatic evaluation of summaries. In *Proceedings of the ACL-04 Workshop on Text Summarization Branches Out (WAS 2004)*, 74–81.

Mani, I., and Maybury, M. 1999. Introduction. In Mani, I., and Maybury, M. T., eds., *Advances in Automatic Text Summarization*. The MIT Press.

Over, P., and Liggett, W. 2002. Introduction to duc-2002: an intrinsic evaluation of generic news text summarization systems. In *Proceedings of the Document Understanding Conference (DUC02)*.

Xie, Z.; Li, X.; Di Eugenio, B.; Xiao, W.; Tirpak, T. M.; and Nelson, P. C. 2004. Using gene expression programming to construct sentence ranking functions for text summarization. In *Proceedings of the 20th International Conference on Computational Linguistics, COLING-2004*, 1381–1384.