Multi-Document Summarization by Maximizing Informative Content-Words

Wen-tau Yih Joshua Goodman Lucy Vanderwende Hisami Suzuki
Microsoft Research
Redmond, WA 98052, USA
{scottyih, joshuago, lucyv, hisamis}@microsoft.com

Abstract

We show that a simple procedure based on maximizing the number of informative content-words can produce some of the best reported results for multi-document summarization. We first assign a score to each term in the document cluster, using only frequency and position information, and then we find the set of sentences in the document cluster that maximizes the sum of these scores, subject to length constraints. Our overall results are the best reported on the DUC-2004 summarization task for the ROUGE-1 score, and are the best, but not statistically significantly different from the best system in MSE-2005. Our system is also substantially simpler than the previous best system.

1 Introduction

In this paper, we show that a very simple approach to generic multi-document summarization can lead to some of the best reported results on this task. Our system has two components: one component uses machine learning to compute scores for each word in the set of documents to be summarized (which is called the "document cluster"); the other component uses a search algorithm to find the best set of sentences from the document cluster for maximizing the scores. Despite the simplicity of the techniques, our results are among the best ever reported for multi-document summarization.

Multi-document summarization is an increasingly important task: as document collections grow larger, there is a greater need to summarize these documents to help users quickly find either the most important information overall (generic summarization) or the most relevant information to the user (topic-focused summarization). Examples where multi-document summarization might be helpful include news, email threads, blogs, reviews, and search results.

The design of our system is motivated by SumBasic [Nenkova and Vanderwende, 2005; Nenkova *et al.*, 2006]. SumBasic is extremely simple, but its performance is within statistical noise of the best system of DUC-2004 and the second best system of MSE-2005, using the ROUGE-1 measure. SumBasic first computes the probability of each content-word (i.e., verbs, nouns, adjectives and numbers) by simply counting its frequency in the document set. Each sentence is scored as the average of the probabilities of the words in it. The summary is then generated through a simple greedy search algorithm: it iteratively selects the sentence with the highest-scoring content-word, breaking ties by using the average score of the sentences. This continues until the maximum summary length has been reached. In order not to select the same or similar sentence multiple times, SumBasic updates probabilities of the words in the selected sentence by squaring them, modeling the likelihood of a word occurring twice in a summary.

Our system improves on SumBasic in three ways. First, in contrast to SumBasic which only uses frequency information, our approach also uses position information. Second, we introduce a discriminative, machine-learning based algorithm to combine these information sources. Third, instead of applying the heuristic greedy search of SumBasic, we formalize the content selection process as an optimization problem. Specifically, we use a stack-decoding algorithm to find the summary that maximize the total scores of the content-words it has, subject to the summary length constraint. Each of these three improvements is empirically shown to lead to better summaries. This system achieves the best reported ROUGE-1 results for the DUC-2004 summarization task as well as for MSE-2005, although the difference is statistically significant only for DUC 2004. (Another recent system [Wan and Yang, 2006] also reports excellent results on the DUC-2004 task, but uses a different version of the Rouge software, making comparisons difficult.)

The rest of this paper is organized as follows. In Section 2 we describe how we determined the importance of position information, and then we describe several term scoring mechanisms. Next, in Section 3, we give details of our optimization technique, including the stack decoder. In Section 4, we give experimental results showing that both our new scoring method and our optimization technique lead to improvements, and that the combined system outperforms the best previous system on the ROUGE-1 metric. We then compare our system to related work in Section 5, and finally Section 6 concludes the paper.

2 Scoring

In this section, we describe our techniques for scoring words. Our starting point, SumBasic, scores words purely using frequency information. In Section 2.1, we investigate additional

possible features by looking for mismatches between human and machine-generated summaries. We identify position information as a key additional feature. Then, in Section 2.2, we examine two different ways to combine position information with frequency information, one of which can be described as *generative*, and the other of which as *discriminative*.

2.1 Position and Frequency Features

As mentioned previously, exclusively using word frequency information, SumBasic can produce quite competitive multi-document summaries [Nenkova *et al.*, 2006]. They focused on frequency information after comparing machine to human generated summaries, and finding that machine-generated summaries contained fewer frequent words than human generated ones. We wanted to build on this work by identifying other information sources that could be useful for a summarization system, in addition to word frequencies. In particular, we examined the summaries generated by SumBasic and human summaries written for the same document set, and checked whether these two types of summaries had different properties. For example, if the number of capitalized words was substantially higher (or lower) in human summaries than in machine generated summaries, we would expect that capitalization information could help improve the automatic system. We looked at a number of different properties, including capitalization, word length, sentence length, and others. We found that the human and machine summaries had comparable values for all properties except one – the word positions.

Position information has been used quite frequently in single-document summarization. Indeed, a simple baseline system that takes the first l sentences as the summary outperforms most summarization systems in the annual DUC evaluation [Barzilay and Lee, 2004] (and see also [Zajic *et al.*, 2002] for use of position in scoring candidate summary sentences). For multi-document summarization in DUC, position information has also been used as part of simple baseline systems: two baselines were constructed for multi-document summarization, both informed by position. One system took the first n words of the most recent news document in the collection as the baseline, while the other system was constructed by appending the first sentences of subsequent articles in the document cluster until the length limit was reached.

In mutli-document summarization, various systems have used sentence position as a feature in scoring candidate sentences (e.g., [Radev *et al.*, 2001; Zajic *et al.*, 2005]), but word position has not been explored thus far. At the level of words, some systems have used as a feature the number of words in the candidate sentence that are found to be *signature tokens*, i.e., words found to be frequent near the beginning of an article and therefore likely to be highly informative [Lin and Hovy, 2000; McKeown *et al.*, 2001; Schiffman, 2002; Conroy *et al.*, 2004]. However, these signature tokens are computed based on a large corpus of news articles, not based on the word position in a small cluster of articles.

To check for the importance of word position information in a given cluster of documents, we first needed to define a position measure. Our procedure was as follows: for each word occurrence in a document cluster, we computed its position relative to the beginning of its document, e.g. 0 for the first word and 1 for the last. For each word, we then computed the average of its occurrences throughout the document cluster. We call this the average position of the word. A very frequent word occurring randomly throughout the documents would have an average position of about 0.5, but words that occur disproportionately at the beginnings of documents have an average position below 0.5.

In particular, we compute the average position in the original document cluster for the terms in the summary. For human summaries, the value is about 0.42. For term frequency-based summaries generated by SumBasic, the value is around 0.46. Compared to the value computed from the document cluster, which is 0.5, this fact re-confirms the importance of position. Of course, a difference of .04 may or may not be important in practice: later, in Section 4, we will give experimental results showing that our inclusion of this information does indeed lead to statistically significant improvements.

2.2 Scoring the terms

Knowing that frequency and location are important, the next question is how to incorporate them together to create a good scoring function for each term. In this section, we explore two different approaches: *generative* and *discriminative*.

Generative Scoring

The term scoring method used in SumBasic can be thought of as a generative model based on frequency alone. In this model, we assume that summaries are generated at random from the document cluster according to the following process. We select a term at random from the document cluster, with a probability proportional to the frequency of the word in the cluster. We add this term to the summary, and then delete all occurrences of the term from the cluster. We repeat this process until we have generated a summary of the required length. Of course, a random set of words would be a horrible summary. In practice, the output summary consists of sentences with the highest probability according to the model, selected from the document cluster. In Section 3, we describe in detail how we use these scores to create summaries.

In addition to the above model based only on frequency, we also propose a generative model that favors words from the beginnings of documents. We tried additional experiments in which, instead of selecting words at random from the whole document, we select terms only from the very beginning of the document. For our experiments with DUC-2004, we used the first 100 words; for MSE-2005, where many of the articles were very short, we used the first 50 words.

Our final generative model combines the above two models, allowing a tradeoff between overall frequency, and frequency at the beginning of documents. This model assumes that the first step is to flip a biased coin. Based on the biased coin flip, we either select our terms at random from the whole document cluster, or only from the beginnings of the document clusters. The effective probability value is therefore a linear interpolation of the values used in the two base models. After trying different bias terms, we found that setting it to 0.5 worked best empirically.

Discriminative Scoring

Often, it has been found that for probabilistic approaches to natural language processing, discriminative approaches work better than generative ones. We train our discriminative model using the data for the DUC-2003 multi-document summarization task. That is, we *learn* the probability that a given term in the document will be in the summary. For each content word in a document cluster, we assign label 1 to it if it appears in the given human summary; otherwise, the label is 0. We then try to learn the probability that the term has label 1, given its features.

The learning algorithm we chose is logistic regression, for two reasons. First, logistic regression predicts probabilities directly (in contrast to, say, perceptron or SVMs). Second, logistic regression works well even with inconsistent labels (in contrast, to, say, large margin approaches like SVMs which can find such data difficult to train on). This is important since we have four different human summaries for each document cluster. When a term in the document cluster only appears in, say, three summaries, the training data will have four identical feature vectors representing this term. Three of them will be labeled 1 and the other one will be labeled 0.

We created 7 basic features using the frequency and position information. They are: #occr (the number of the occurrences in the document cluster), occrRatio (#occr divided by the number of the content-words in the document cluster), avgOccrRatio (calculate occrRatio for each document in the cluster, and average them), minPos (find the position (starting from 0) where the content-word appears first in each document in the cluster, and use the smallest), avgMinPos (similar to minPos, but use the average instead; if the content-word does not appear in a document, the position is the total number of content-words in this document), minRelPos (compute the relative first position, which is the minPos divided by the number of content-words in the document, and return the smallest of the cluster), and avgRelPos (similar to minRelPos, but use the avearge instead).

For each of the basic features, we also create a corresponding *log* feature. Suppose the basic feature is x; then the corresponding log feature is $\log(1 + x)$. We then expand the feature space by introducing conjunction features. For each pair of the above features, we use the product of the values as a new feature, which has a similar effect to using a degree-2 polynomial kernel.

3 Optimization Method

One relatively novel aspect of our system is that we have moved from an algorithmic description common in most systems, to a scoring description: potential summaries are given an overall score based on the scores of the included content words, and the goal is to find the summary with the best overall score. We first explain how we decide the overall score of a summary in Section 3.1. Then, in Section 3.2, we describe the optimization procedure that searches for the best summary. Finally, in Sec 3.3, we describe a variation that does sentence simplification, to allow the system to use either full sentences, or sentences with certain less useful phrases removed.

3.1 Scoring the summary

We considered two different methods for combining scores of words to get an overall score: a product-based method, and a sum-based method. Consider the generative model of Section 2.2. For a given summary, we could multiply together the probabilities from this model. Finding the summary with the highest product of probabilities would give us the summary with the highest probability of being exactly correct, according to the generative model.

On the other hand, in automatic evaluation metrics such as the ROUGE scores, we favor summaries that have the most words that also appear in the reference (i.e., human) summaries. If the score of a content word represents the probability that the word appears in the human summary, then the summation of these scores can be thought of as the expected number of "correct" content words. We compared the sum and product methods, and we found that the sum method consistently worked better.

Our summary scoring principle is somewhat different from SumBasic in that we directly compute the score based on the words, while SumBasic weights sentences first and tries to select sentences which have a better total score.

3.2 Finding the *best* summary

The iterative algorithm used in SumBasic can be thought of as a simple greedy algorithm that tries to find the best summary. However, this greedy algorithm rarely finds the best summary, even in the sense of optimizing the expected score. In addition, since it does not explicitly consider the max-length cut-off threshold, which causes the end of the last sentence not to be used for scoring, the score of the final summary may be further impacted. We thus developed a more complex algorithm that could explicitly search for the best combination of sentences. Our algorithm is based on a stack decoder [Jelinek, 1969]. One typical problem of stack decoders is that they have trouble comparing hypotheses of different lengths. Although it is sometimes solved with an A* search [Paul, 1991], this requires finding an admissible cost function, which does not always exist. Instead of using an A* search, we chose to use multiple stacks, with each stack representing hypotheses of different lengths [Magerman, 1994].

Our stack decoder method is shown in Algorithm 1, which takes as input the set of all sentences from the document cluster, as well as the *score* array used to weight the sentences. The method uses *maxlength* stacks: one for each length, up to the maximum length of the summary. Each stack contains our best summaries so far, of exactly that length. (The last stack, *stack*[*maxlength*], may contain summaries longer than *maxlength* – but words past size *maxlength* are not considered as part of the scoring). There will be at most *stacksize* different hypotheses on any given stack.

The algorithm proceeds by examining a particular stack. It looks at every solution on that stack (a solution is a set of sentences). It then tries to extend that solution with every sentence from the document cluster. These extensions are then placed on the stack of the appropriate length. In order to avoid an exponential blowup in the number of solutions on any given stack, we use a priority queue, and only keep the top *stacksize* highest scoring solutions on any given stack.

Algorithm 1 Stack Decoder for Multi-Document Summarization

```
INPUT: An array of Sentences[] and scores for each term in the Sentences[]
INPUT: A maximum length maxlength
INPUT: A maximum stacksize
TYPEDEF Solution = A variable length array of sentence IDs
Let stack[0..maxlength] be a priority queue of Solutions; each queue has at most stacksize Solutions.
stack[0] = the Solution of length 0;
for i = 0 to maxlength − 1 do
  for all sol ∈ Stack[i] do
    for all s ∈ Sentences do
      newlen = min(i+length(s),maxlength)
      newsol = sol ∪ {s}
      score = score of newsol counting each word once, and at most maxlength words
      Insert newsol, score into queue stack[newlen], pruning if necessary
    end for
  end for
end for
Return best scoring solution in stack[maxlength]
```

Notice that if we did not penalize words that occur more than once, and if we did not truncate the very last sentence as part of the scoring procedure, then this problem would be equivalent to the Knapsack Problem: how large a score can we pack into a *maxlength* word summary. Without the no-duplication limitation and last sentence truncation, and when using a stack size of 1, Algorithm 1 devolves to the standard exact solution using dynamic programming for the Knapsack Problem.

In Section 4 we will compare the greedy algorithm in SumBasic to the stack decoder algorithm, and show that the stack decoder leads to reasonably large gains. Note that this algorithm is fast: about 11 seconds per document cluster with a stack size of 30 on a standard PC.

3.3 Summarization with simplified sentences

The goal of a summarization system is to produce summaries with as much content as possible given the length limit. Therefore, if we can *simplify* sentences in the summary, removing phrases that have little or no expected value, we can make room for additional sentences that provide more value.

For each sentence in the document cluster, our sentence simplification procedure eliminates various syntactic units based on predefined heuristic templates, such as removing noun appositives, gerundive clauses, nonrestrictive relative clauses, or intra-sentential attributions. Unlike previous approaches that deterministically shorten sentences before or after sentence selection (e.g., [Conroy *et al.*, 2005; Siddharthan *et al.*, 2004; Daumé III and Marcu, 2005]), the simplified sentence in our approach does not replace the original sentence but is instead added to the sentence pool for the summarizer to choose from. The choice among the sentence alternatives provided by the simplification procedure is left entirely to the summarization component. A detailed description of our approach to sentence simplification can be found in [Vanderwende *et al.*, 2006].

4 Experiments

In order to evaluate the performance of our systems, we use two data sets that have been used in recent multi-document summarization shared tasks: multi-document summarization (task 2) in DUC-2004 and the multilingual multi-document summarization task in MSE-2005. We first show the results of the purely extractive system on each of these tasks, and also show the effects of variations of the systems. Next, we perform experiments using the sentence simplification system, showing additional improvements.

DUC 2004

In the multi-document summarization task in DUC-2004, participants are given 50 document clusters, where each cluster has 10 news articles discussing the same topic, and are asked to generate summaries of at most 100 words for each cluster. Since the same task was also held in DUC-2003, but with different documents, we take the 2003 data for development, especially for training the probabilities.

We present the results of our system and SumBasic using different term scoring methods in Table 1. In addition, we also compare them with the best system (peer65) and the baseline system (greedyline) in DUC-2004. As mentioned previously, *greedyline* simply takes the first 100 words of the most recent news article in the document cluster as the summary. For the evaluation, we use the ROUGE-1 metric (with stemming and stop-words removed), which has been shown to correlate well with human judgments [Lin and Hovy, 2003; Lin, 2004] and which was found to have one of the best correlations with human judgments on the DUC-2004 data [Over and Yen, 2004]. In addition, we also report the performance on ROUGE-2 (bigram overlap) and ROUGE-SU4 (skip bigram) metrics[1].

In the table, the *stack* systems use our stack decoder algorithm, while the *basic* systems use SumBasic's iterative algorithm. The suffixes of the system names indicate the types of term scoring methods used. Discriminative training is represented by *-train*; using frequencies only is denoted as *-freq*; using frequencies in the first 100 words is *-pos*; and finally, *-inter* means the score is the average of full document frequencies and frequencies in the first 100 words.

We performed paired t-tests comparing our systems to peer65 (the previous best performing system) and to SumBasic (basic-freq in our terminology). The top three systems (*stack-train*, *stack-inter*, and *basic-inter*) were all significantly better on ROUGE-1 than peer65 ($p < .05$). On ROUGE-2 and ROUGE-SU4, these three systems were just slightly worse than peer-65, but the differences were not significant. The top four systems were all significantly better than basic-freq ($p < .01$). We have thus improved on both our baseline system and on the best previous system.

[1] ROUGE version 1.5.5, with arguments -a -n 4 -w 1.2 -m -2 4 -u -c 95 -r 1000 -f A -p 0.5 -t 0

SYSTEM	ROUGE-1	ROUGE-2	ROUGE-SU4
stack-train	0.327	0.086	0.129
stack-inter	0.322	0.086	0.129
basic-inter	0.322	0.088	0.132
basic-train	0.320	0.084	0.128
stack-freq	0.311	0.076	0.119
stack-pos	0.310	0.082	0.126
basic-pos	0.306	0.082	0.126
peer 65	0.305	0.090	0.131
basic-freq	0.303	0.077	0.121
greedyline	0.202	0.061	0.099

Table 1: DUC-04: ROUGE-1 (stop-words removed and stemmed), ROUGE-2 and ROUGE-SU4 (stemmed) scores

SYSTEM	ROUGE-1	ROUGE-2	ROUGE-SU4
stack-inter	0.378	0.140	0.173
stack-pos	0.375	0.134	0.166
basic-train	0.374	0.136	0.173
stack-freq	0.374	0.129	0.166
stack-train	0.369	0.133	0.167
basic-inter	0.367	0.133	0.161
peer 28	0.365	0.160	0.186
basic-pos	0.362	0.131	0.163
basic-freq	0.352	0.103	0.161

Table 2: MSE-05: ROUGE-1 (stop-words removed and stemmed), ROUGE-2 and ROUGE-SU4 (stemmed) scores

We also looked at the component improvements. In every case, stack decoding was better than basic (greedy) decoding, but the differences were not statistically significant. Differences between a pure frequency approach and the trained approach (*stack-train* versus *stack-freq* and *basic-train* versus *basic-freq*) were both highly significant ($p < .01$). The difference between *basic-inter* and *basic-freq* was highly significant ($p < .01$), showing that using position information does indeed lead to improvement.

MSE 2005

In 2005, a different multi-document summarization task was conducted as part of the Machine Translation and Summarization Workshop at ACL. Participating systems produced a 100-word summary from a document cluster, which was a mixture of English and Arabic news articles on the same topic, where the Arabic documents are translated into English by automatic machine translation systems. In addition to this major difference, the news articles are generally shorter than those used in DUC-2004. Ignoring the potential mistakes introduced by the machine translator, we ran our systems without specific modifications for this unusual setting, except counting the frequencies of the first 50 words instead of 100 in our position-based generative models (*-pos* and *-inter*).

As shown in Table 2, on this data set, the mean ROUGE-1 score of our best system, *stack-inter*, is better than the best participating system (peer 28) and the original version of SumBasic. On ROUGE-2 and ROUGE-SU4, the scores of our systems are slightly lower. However, in all three metrics, all of the systems have no statistically significant difference.

We notice that the our stack-decoding summarizer with the discriminatively trained term scores does not perform the best. This may be due to the fact the model is trained on the DUC-2003 data, which may be quite different from the data in MSE-2005.

SYSTEM	ROUGE-1	ROUGE-2	ROUGE-SU4
stack-train-sim	0.339 (+0.012)	0.086	0.129
basic-train-sim	0.328 (+0.008)	0.084	0.130
stack-freq-sim	0.320 (+0.009)	0.077	0.120
basic-freq-sim	0.312 (+0.009)	0.075	0.121
stack-train	0.327	0.086	0.129
basic-train	0.320	0.084	0.128
stack-freq	0.311	0.076	0.119
basic-freq	0.303	0.077	0.121

Table 3: DUC-04: ROUGE-1, ROUGE-2 and ROUGE-SU4 scores, with sentence simplification

Sentence simplification for DUC 2004 and MSE 2005

Next, we look at the effects of sentence simplification. Table 3 shows the performance of different configurations of our summarizer with sentence simplification (-sim), and without; the number in parentheses for the ROUGE-1 score is the amount of improvement. Comparing the four pairs of configurations (e.g. *stack-train-sim* versus *stack-train*, etc.), we see that sentence simplification consistently raises the ROUGE-1 score by about .01, while having essentially no impact on ROUGE-2 or ROUGE-SU4. The difference in ROUGE-1 score was largest for *stack-train-sim* versus *stack-train*, and the difference was statistically significant ($p < .05$) for both this pair, and *stack-freq-sim* versus *stack-freq*. There appears to be a synergy to using our stack decoder with sentence simplification: the stack decoder allows the system to do a better job of choosing among the many candidate sentences, which include both all of the original sentences, and simplified versions of many sentences.

5 Related Work

In this section, we compare our work to related work on a number of aspects, such as scoring method, search method, etc.

Both SumBasic and our system focus on scoring individual words. In contrast, most existing system are sentence-based. These sentence-based systems use a variety of features, including: sentence position in the document, sentence length, sentence similarity to previously extracted sentences (usually using the maximal marginal relevance (MMR) framework [Carbonell and Goldstein, 1998]), an explicit redundancy score [Daumé III and Marcu, 2005], and sentence similarity to the document centroid. In the cases where individual words are considered during sentence selection, important words are identified through graph-based analysis where the nodes in the graph represent words [Mani and Bloedorn, 1997; Erkan and Radev, 2004], rather than through probabilistic measures such as those used in this work. In contrast to these complex systems, the only features we use are frequency and position-based.

We combine the position and frequency information using either a simple generative or simple discriminative

model. Other summarization systems use heuristic methods for combining multiple features, or, as in [Daumé III and Marcu, 2005], have trained parameters directly using various ROUGE metrics as an objective function.

Almost all previous multi-document summarization systems, including SumBasic, have used greedy or heuristic searches to choose which sentences to use, even when they had an explicit scoring function. In this paper, we formalize summarization as an optimization problem and present an explicit search algorithm, namely a stack decoder, to search for the best combination of sentences.

A complete survey of summarization systems is beyond the scope of this paper, but it's worth describing in more detail the CLASSY summarization system, which was the previous best system on the ROUGE-1 metric for the DUC-2004 task (peer 65), and is now tied for best for the MSE-2005 task (peer 28). The CLASSY [Conroy *et al.*, 2004; 2005] summarization system consists of two core components – a Hidden Markov Model for selecting sentences from each document and a pivoted QR algorithm for generating a multi-document summary. The HMM has two kinds of states, which correspond to summary and non-summary sentences in a single document. The model uses just one feature, which is the number of signature terms in each sentence. These terms are decided by the log-likelihood statistic suggested by Lin and Hovy [2000], derived based on a large set of documents in advance. In addition, the best number of the HMM states needs to be determined based on empirical testing, and the HMM model needs to be learned using training data. After applying the HMM, the top scoring sentences of each document form a weighted token-sentence matrix. A pivoted QR algorithm is then used for scoring and selecting sentences to form the output summary. In addition to these two core components, CLASSY also incorporates a linguistic component as a preprocessing stage to provide the summarization engine simplified (shortened) sentences as input.

Very recently, Wan and Yang (2006) proposed an approach to multi-document summarization based on affinity graphs. Their method tried to identify semantic relationships between sentences and used a graph rank algorithm to compute the amount of information a subset of sentences contain. The best subset of sentences were then selected as the output summary using a greedy algorithm. Their system was also very competitive and outperformed the best result in DUC-2004. However, because of the different ROUGE versions and parameter settings, we are not able to compare directly with their results.

6 Conclusion

Our results are the best reported on the DUC-2004 and MSE-05 multi-document summarization tasks for the ROUGE-1 score, although only for the DUC-2004 task is the difference statistically significant, and we could not compare to Wan and Yang (2006) because of the different ROUGE versions used. On ROUGE-2 and ROUGE-SU4, our system is second best, although the differences are not statistically significant. We have achieved these excellent results using a system that is substantially simpler than the previous best system, CLASSY. In particular, our system used about 200 lines of code for the stack decoder, and less than 400 lines for the score computations, as well as pre-existing libraries for logistic regression and for finding content words.

We achieved these results by enhancing an already fairly competitive summarization system, SumBasic, in several aspects. First, we showed that position information was not already sufficiently captured by SumBasic. Second, we proposed different word scoring methods that combine the position information with frequency information. We gave a very easy-to-implement generative model that produces excellent results, and we described a somewhat more complex, but still straightforward, discriminatively-trained version that works even better. While a few other systems have explicitly optimized parameters [Daumé III and Marcu, 2005; Erkan and Radev, 2004], we are not aware of any previous work optimizing for word scores. Third, we described a simple search procedure using a stack decoder that can find the best sentences to form a summary, given the word scores. In contrast to more common approaches using heuristic or greedy methods, such as the iterative algorithm of SumBasic, the explicit search method is not only more principled and with a clear objective function, but also better empirically.

As for future research, we would like to apply our method also in single-document summarization. Given that position plays an even more important role in this task, we believe our system should be able to perform reasonably well and unify single-document and multi-document summarization tasks seamlessly. While frequency may be less informative given that there is less repetition, our discriminative model provides a way to incorporate other information that can be helpful in judging word importance, especially in the single-document summarization setting. Another area we would like to explore is to enhance the readability of the summary, an important issues existing in almost all purely extractive summarization systems. While the summary generated by our system is very informative, the coherence between sentences in the summary may be further enhanced by adjusting the order better or applying some semantic analysis to reconstruct the summary. Finally, we would like to adapt our system to task-focused summarization problems, such as web search result snippets.

Acknowledgments

We thank Arul Menezes, who helped us on the initial setting of the experiments. We are also grateful to anonymous reviewers for their valuable comments.

References

[Barzilay and Lee, 2004] R. Barzilay and L. Lee. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL*, 2004.

[Carbonell and Goldstein, 1998] J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, 1998.

[Conroy *et al.*, 2004] J. Conroy, J. Schlesinger, J. Goldstein, and D. O'Leary. Left-brain/right-brain multi-document summarization. In *Proc. of DUC*, 2004.

[Conroy *et al.*, 2005] J. Conroy, J. Schlesinger, and J. Goldstein. Three classy ways to perform arabic and english multi-document summarization. In *Proc. of MSE*, 2005.

[Daumé III and Marcu, 2005] H. Daumé III and D. Marcu. Bayesian multi-document summarization at MSE. In *Proc. of MSE*, 2005.

[Erkan and Radev, 2004] G. Erkan and D. Radev. LexPageRank: Prestige in multi-document text summarization. In *Proc. of EMNLP*, 2004.

[Jelinek, 1969] F. Jelinek. Fast sequential decoding algorithm using a stack. *IBM Journal of Research and Development*, 1969.

[Lin and Hovy, 2000] C. Lin and E. Hovy. The automatic acquistion of topic signatures for text summarization. In *Proc. of COLING*, 2000.

[Lin and Hovy, 2003] C. Lin and E. Hovy. Automatic evaluation of summaries using n-gram co-occurance statistics. In *Proc. of HLT-NAACL*, 2003.

[Lin, 2004] C. Lin. ROUGE: a package for automatic evaluation of summaries. In *Proc. of the Workshop in Text Summarization, ACL 2004*, 2004.

[Magerman, 1994] David Magerman. *Natural Language Parsing as Statistical Pattern Recognition*. PhD thesis, Stanford University University, February 1994.

[Mani and Bloedorn, 1997] I. Mani and E. Bloedorn. Multi-document summarization by graph search and matching. In *Proc. of AAAI*, 1997.

[McKeown *et al.*, 2001] K. McKeown, R. Barzilay, D. Evans, V. Hatzivassiloglou, M-Y. Kan, B. Schmiffman, and S. Teufel. Columbia multi-document summarization: Approach and evaluation. In *Proc. of DUC*, 2001.

[Nenkova and Vanderwende, 2005] A. Nenkova and L. Vanderwende. The impact of frequency on summarization. Technical report, MSR-TR-2005-101, 2005.

[Nenkova *et al.*, 2006] A. Nenkova, L. Vanderwende, and K. McKeown. A compositional context sensitive multi-document summarizer. In *Proc. of SIGIR*, 2006.

[Over and Yen, 2004] P. Over and J. Yen. An introduction to DUC 2004 intrinsic evaluation of generic news text summarization systems. In *Proc. of DUC*, 2004.

[Paul, 1991] D. Paul. An efficient A$*$ stack decoder algorithm for continuous speech recognition with a stochastic language model. In *ICASSP*, 1991.

[Radev *et al.*, 2001] D. Radev, S. Blair-Goldensohn, and Z. Zhang. Experiments in single and multi-document summarization using MEAD. In *Proc. of DUC*, 2001.

[Schiffman, 2002] B. Schiffman. Building a resource for evaluating the importance of sentences. In *Proc. of LREC*, 2002.

[Siddharthan *et al.*, 2004] A. Siddharthan, A. Nenkova, and K. McKeown. Syntactic simplification for improving content selection in multi-document summarization. In *Proc. of COLING*, 2004.

[Vanderwende *et al.*, 2006] L. Vanderwende, H. Suzuki, and C. Brockett. Microsoft Research at DUC2006: Task-focused summarization with sentence simplification and lexical expansion. In *Proc. of DUC*, 2006.

[Wan and Yang, 2006] X. Wan and J. Yang. Improved affinity graph based multi-document summarization. In *Proceedings of HLT-NAACL, Companion Volume: Short Papers*, pages 181–184, 2006.

[Zajic *et al.*, 2002] D. Zajic, B. Dorr, and R. Schwartz. Automatic headline generation for newspaper stories. In *Proc. of DUC*, 2002.

[Zajic *et al.*, 2005] D. Zajic, B. Dorr, J. Lin, C. Monz, and R. Schwartz. A sentence-trimming approach to multi-document summarization. In *Proc. of DUC*, 2005.