CIS 530 Final Project: Automatic Summarization Systems

Jingyi Wu
wujingyi@seas.upenn.edu

Emily Boggs
emboggs@seas.upenn.edu

Abstract

For this project, we have implemented three basic summarizers: a TF*IDF summarizer, a LexRank summarizer, and a KL-divergence summarizer. In addition, we have designed a new summarizer that ranks sentences using a feature-based classifier.

1 General Parameters

Some system parameters are consistent for all of the summarizers:

- Each summarizer takes two parameters: an input collection with groups of documents to summarize, and the output folder in which to save the summaries.
- The summarizer output is named by prefixing the name of the current subdirectory with "sum_". For instance, the input collection subdirectory "dev_00" would produce the summary "sum_dev_00.txt".
- The sentences of the input collections are always lowercased.
- Only sentences that are between 9 and 45 words long are considered for inclusion in the summary.
- To reduce redundancy, sentences that exceed a certain threshold of similarity with any sentence already in the summary are not added. The exact value of the threshold differs between implementations.
- For TF*IDF calculations, IDF was computed from the New York Times corpus. Because this corpus does not contain every word in the test inputs, IDF calculations used add-one smoothing.

2 Basic Systems

2.1 TF*IDF System

The TF*IDF Summarizer ranks sentences by calculating the average TF*IDF score for the words in each sentence. In this implementation, stopwords are included in the calculation, because the proportion of stopwords to content-ful words is an important aspect of the TF*IDF ranking. The sentence score is calculated over all word types, instead of all tokens. In tests with the development data, using types instead of tokens was found to result in a small increase in ROUGE-2 recall. In reducing redundancy, the best similarity threshold was found to be 0.8, which resulted in the highest ROUGE-2 recall for the development data.

2.2 LexRank System

The LexPageRank Summarizer represents each sentence as a node in the graph. Sentences are connected through an edge if they have a cosine similarity of TF-IDF over the threshold 0.1, which we found yields best results. The graph is represented as a matrix normalized in columns. We used power iteration to calculate the prime eigenvector for each matrix. The stopping criteria for the iteration is when the square root of sum of squares of the differences between the pre-iteration and post-iteration value of each node falls to/under

0.15 ($\sqrt{d_1^2 + d_2^2 + ... + d_n^2} <= 0.1$). According to our experiments, this criteria renders good ROUGE scores. The resulting prime eigenvector contains the final score of each sentence. Ranking the sentences according to the score, we generate a summary containing top-ranking sentences.

2.3 KL Divergence System

The KL Summarizer selects sentences to add to the summary by greedily choosing the sentence that will minimize KL divergence between the summary and the input. For this implementation, stopwords are ignored in all calculations. Because unigram distribution of stopwords is less likely to be substantially different between the summary and the input, removing them would highlight the meaningful words in the distribution. No smoothing was performed in calculating the distributions; since KL is a sum of probability calculations, a zero term does not significantly affect the output. The threshold for removing redundant sentences is set to 1, which means that sentences are not ignored on the basis of similarity to sentences already in the summary. This value was chosen based on tests with development data; one possible explanation for the result is that the KL computation takes care of redundancy issues itself, causing any further effort to be counterproductive.

2.4 Performance on Development Set

System	TF*IDF	LexRank	KL
Rouge-2 Recall	0.08058	0.07194	0.08899

3 Our Summarization System: FeatureSum

3.1 System Design

Our summarizer is designed around the idea that there are certain features that are distinctive of good summaries. Summaries should address the main points of the documents being summarized. In addition, human-written summaries tend to be general rather than specific. At first, we used these and other intuitions about summaries to build features for a classifier that would rank the sentences using SVM-Rank. However, this system was ineffective, with a prohibitively long runtime and a subpar automatic annotation component. To avoid these issues, we pared down the list of features and removed the need for supervised learning, creating a method of scoring sentences that does not require weighting or normalizing of features.

The scoring mechanism of the summarizer uses three features from the classifier: specificity, topic words, and sentence position. These features were chosen because their relationship to the notion of a "good" summary is fairly predictable, whereas the effect other features (such as named entities and word polarity) was less intuitive. For descriptions of the calculation of feature values, see the following section.

In the summarizer, sentences are ranked in Python's implementation of a priority queue, which returns elements in order from smallest to greatest priority. Therefore, the scores needed to be generated such that a lower score means that a sentence is more likely to be included in the summary. To achieve this, we first analyzed whether each feature should be large or small for a good sentence. If a sentence is general, it will have a small specificity value, since this value is basedon hypernym distance. The count of topic words should be high, as should the weight given to the sentence for its position. Thus, to minimize the score of a good sentence, the specificity is divided by the topic word count and the position weight : $specificity/(topicwords * position)$. In other words, the best sentences are the ones that do a good job of minimizing specificity while maximizing topic words and position.

3.2 Feature Calculation and Resources

WordNet is used to determine sentence specificity. The specificity of each word in a sentence is computed as the distance between the word and the root hypernym in WordNet. Each word is designated as specific, general, or medium according to predetermined thresholds. The specificity of the sentence is the average specificity of the nouns in that sentence. The motivation for this feature is the idea that summaries might be more likely to contain general terms instead of specific ones.

TopicS is used to calculate the number of topic words for the collection that are in a sentence. Sentences that contain topic words are likely to be discussing the general topic of an article, as opposed

to a specific detail; these sentences should be more highly ranked. In order to more accurately identify topic sentences, we stemmed both topic words and words in testing input.

Sentence position was defined as whether a sentence was at the beginning of a document; the first sentence is given the highest weight, the second sentence is given a lower weight, and all other sentences are given a very low weight. The values of these weights was determined by manual testing of different weight proportions.

3.3 Performance

On the development data, this system achieves a ROUGE-2 recall of 0.09221.

4 Discussion and Analysis

One of our criteria of choosing summary sentences is whether the sentence contains enough topic words. Therefore, our summarizer outperforms the three basic summarizers above, because by using TopicS we give such sentences an advantage over sentences that are not related to the topic. We experimented with stemming both topic words and testing input, and found stemming not helping in this case, with a ROUGE-2 recall at around 0.086, compared to the score for non-stemming summarizer at 0.090. This may be because the topic words in the input data are mostly very specific nouns that does not have many derived word forms. Another possible reason is that the topic words only appear in the documents in the same form, even though it may have many derived word forms. On the other hand, stemming may result in ambiguation of the specificity of a word, resulting in giving advantage to words that have similar forms (e.g. same root) with the topic, but are not closely related to the topic. In our summarizer, this tradeoff probably offsets the advantage of stemming.

References

Gunes Erkan and Dragomir R. Radev. *LexPageRank: prestige in multi-document text summarization.* Proceedings of EMNLP2004.

Chin-Yew Lin. *ROUGE: A package for automatic evaluation of summaries.* Proceedings of ACL2004.

Annie Louis and Ani Nenkova. *Automatic identification of general and specific sentences by leveraging discourse annotations.* Proceedings of IJCNLP2011.