OGI/OHSU Baseline Query-directed Multi-document Summarization System for DUC-2005

Seeger Fisher, Brian Roark,
Center for Spoken Language Understanding
OGI School of Science & Engineering
Oregon Health & Science University
{roark,fishers}@cslu.ogi.edu

Jianji Yang and Bill Hersh
Department of Medical Informatics &
Clinical Epidemiology
Oregon Health & Science University
{yangj,hersh}@ohsu.edu

Abstract

In this paper, we briefly outline the sentence extraction system that we developed for the 2005 DUC Summarization Evaluation. Training involved learning a sentence ranking model using Support Vector Machines and some simple features. Sentence selection from the ranked list involved picking sentences with the most non stop-word overlap with the query. We also filtered some sentences, both based on sentences already in the summary, and on other sentence-level features. We employed some simple sentence compression techniques, as well.

1 Introduction

This is the first summarization system that the authors have developed, and we intend it to be a simple baseline system that can hopefully be improved over time to be competitive with other systems. At that point, we would like to try to move the system beyond simple extraction with various NLP approaches.

At the highest level, we broke this problem down into two tasks: 1) sentence ranking based on simple features; and 2) sentence selection from a ranked list. For sentence ranking, we used SVMlight (`http://svmlight.joachims.org/`), which can be parameterized to learn a preference ranking. For sentence selection we used a simple overlap metric between the query and candidate sentence to only select sentences that were relevant to the query, since the SVM ranking does not take the query into account.

In the next section, we will first present the training and application of the sentence ranking approach. Next, we will present sentence selection from the ranked list.

2 Sentence extraction system

2.1 Sentence ranking

For every cluster of documents c in the set of clusters $\mathcal{C}$ comprising the training set, let $\mathcal{Z}_c$ be the collection of manual summaries for that cluster. Let $s \in c$ be the sentences in cluster c and $z \in Z_c$ be the sentences in the summaries of cluster c. For every cluster $c \in \mathcal{C}$ we scored each sentence $s \in c$ as follows

$$\rho(s) \quad = \quad \underset{z \in Z_k}{\operatorname{argmax}} \left(\operatorname{rouge}(s, z)\right)$$

where rouge(s, z) is the ROUGE score (Lin and Hovy, 2003) of sentence s with z as the reference summary[1]. We calculated this value for all sentences in each cluster of the DUC 2005 training data, giving us our "gold standard" ranking for use in training the system.

For each sentence in a cluster, we extracted a small number of features for classification. Most of these features are aggregated from word-based features. Word-based features were of three varieties: TF*IDF, log likelihood ratio, and log odds ratio statistics. Let $f(wc)$ be the frequency of word w in cluster $c \in \mathcal{C}$. TF*IDF is defined as follows:

$$\text{tf.idf}(wc) \quad = \quad \frac{f(wc)}{|\{c : f(wc) > 0\}|} \tag{1}$$

Let $\bar{w}$ denote words other than w and $\bar{c}$ denote clusters other than c. Let N be the total word count in our training corpus; $f(w)$ the frequency of the word over all clusters; and $f(c)$ the number of words in cluster c. Then the log likelihood ratio[2] is defined as follows:

$$\text{loglike}(wc) = \log \frac{\alpha}{\beta} \tag{2}$$

where

$$\alpha = f(c)^{f(c)} f(w)^{f(w)} f(\bar{w})^{f(\bar{w})} f(\bar{c})^{f(\bar{c})} \tag{3}$$

[1] For this work, we used the sum of ROUGE-{1-4} and ROUGE-L as our score.

[2] See (Dunning, 1993) for an excellent presentation of the log likelihood ratio statistic.

and

$$\beta = N^N f(wc)^{f(wc)} f(\bar{w}c)^{f(\bar{w}c)} f(w\bar{c})^{f(w\bar{c})} f(\bar{w}\bar{c})^{f(\bar{w}\bar{c})} \quad (4)$$

The log odds ratio[3] is defined as follows:

$$\text{logodds}(wc) = \log \frac{f(wc)f(\bar{w}\bar{c})}{f(\bar{w}c)f(w\bar{c})} \quad (5)$$

For each string we calculate both the average and the sum of all three of these word-based statistics as features. In addition, for the log odds and log likelihood ratios, we calculated the sum of the statistic for just the three highest scoring words in the string. Our ninth and final feature was the position of the sentence in the document.

The reason we used both log likelihood ratios and log odds ratios is that they score co-occurrence dependencies differently. The log likelihood ratio captures whether a word and a cluster occur together at chance or not, and all scores are positive. A high score can indicate that the word and the cluster either occur together surprisingly often or surprisingly rarely. The log odds ratio differs in that it can be positive or negative – positive indicating that the co-occurrence is surprisingly often, negative that it is surprisingly rare, i.e. they are negatively correlated. In addition, the log odds ratio appears to be somewhat more sensitive than the log likelihood ratio to the distribution of relatively infrequent words, i.e. those that occur only a handful of times, which can be quite useful for this task.

To summarize, our feature set consisted of:

1.	average tf.idf	6.	average logodds
2.	sum tf.idf	7.	sum logodds
3.	average loglike	8.	sum (max 3) logodds
4.	sum loglike	9.	Sentence position
5.	sum (max 3) loglike		

For sentence ranking, we used SVMlight (`http://svmlight.joachims.org/`), which can be parameterized to learn a preference ranking, as documented in (Joachims, 2002). Within each document cluster, feature values were normalized into the range [-1,+1], except for the sentence position, which was simply divided by 20 to make the values of the feature of roughly the same order as the other features. We trained our model on an 80 cluster subset of the training corpus. With the learned parameterizations, a ranking of candidate sentences can be obtained from the above mentioned features.

To make word counts robust for feature extraction, we mapped all words to lower case and removed any non-alpha-numeric symbols at word boundaries. Words that occur only once in the corpus were mapped to the "<unk>" symbol. At test time, all non-singleton words in the set of test clusters are added to the word list, and term frequencies and document frequencies are taken over both training and test sets. Note that we did not perform any stemming or removal of stop-words at this stage.

[3]See, e.g., (Agresti, 1996) for a nice presentation of the log odds ratio statistic.

2.2 Sentence selection

We combined the SVM ranking of each sentence with a query relevance for each sentence to get a final ranking. To incorporate the query, we filtered the query of stop words (consisting of all words occurring in the Penn Treebank WSJ sections 02-21 at least 500 times) and stemmed the remaining words. We used the resulting set of stemmed content words from the query to score each sentence as to query relevance, simply by counting the number of stemmed content words in common. We then picked sentences ranked primarily by the query relevance score and secondarily by the SVM ranking.

We selected sentences in order from this final ranking until the summary size limit was reached, with some sentences being filtered. One filter was based on sentence novelty as compared to the summary so far. To check for novelty we again stemmed and removed stop words, this time from the candidate sentence. If the summary already had at least half of the stemmed content-words from the candidate sentence, the sentence was not added. This constraint virtually never occurred in the test sets, though it did in our training data. Our guess is that since the training data sets consisted mostly of articles on the same event, there was a great deal of redundancy compared to the test sets which seemed more often to consist of articles on related events. We also filtered sentences of length less than seven words or greater than 50 words. Sentences starting with a pronoun or quotation mark, or ending in a quotation mark, were rejected.

We performed a crude compression by removing some parentheticals, defined as a string of words within a sentence delimited by dashes, double-dashes or em-dashes. We actually did not filter out strings within parentheses, since these were usually short and contentful in the training data.

Finally, we ordered the extracted sentences by the order in which they occurred in their respective documents, so all first sentences first, then all second sentences, and so on. We believe for this evaluation it would have helped to clump sentences from the same document together; however, our training data consisted mostly of multiple documents that were about the same event, so mixing sentences did not present itself as such an issue until the current evaluation.

3 Evaluation and Future Directions

3.1 Current System Weaknesses

The current summarization system is quite simplistic. Though we scored better than "the middle of the pack" (which was our goal), the system has a number of weaknesses. The basic architecture of our extraction based summarizer seems sound enough: (1) ranking sentences in terms of how likely they are to appear in a summary; (2) combining that ranking with how likely a sentence is to answer the query; (3) choosing sentences in the resulting order, while avoiding repetition, and compressing each sentence when chosen; and (4) re-ordering the sentences. Though we feel this is a sound architecture, each of the pieces is a work in progress.

In the current system, query relevance is purely lexically driven. Though this approach works reasonably well when all documents in a cluster are on similar topics, if the same word is used with a different sense, the current query relevance module will mistake it for a relevant term. This fault would be more problematic if the document clusters were noisy. One approach to solve this problem would be to calculate semantic similarity between a query and a candidate sentence in a more principled way, for instance, using Latent Semantic Analysis (Hofmann, 1999) or Latent Dirichlet Allocation (Blei, et al., 2003). Another approach would be to use query expansion in order to enlarge the number of terms in the query.

Our current sentence compression module is very light-weight, only deleting a subset of parentheticals. There are some promising sentence-compression algorithms (Turner and Charniak, 2005). Augmenting such a compression algorithm with evidence from query terms or semantic similarity ought to provide good compression with little loss of valuable content.

The current system re-orders sentences simply by outputting them in document order. A problem with this approach is that it leads to a lack of cohesion, since sentences from different documents are interleaved. The simplest way to fix this would be to group sentences by document, then by document order. A more sophisticated approach would be to use a discourse-cohesion algorithm to order the sentences.

3.2 Possible Improvements

One area for improvement, at least relative to Pyramid scores, would be to train our sentence ranker on only the ROUGE metrics that best correlate to Pyramid scores. Since we didn't know these ahead of time, our ranker used the average of the ROUGE scores for its objective. We also plan to incorporate some lexical features, with the intuition that some generic words or phrases may often indicate a good candidate for extraction.

The current system automatically filters candidate sentences that start with a pronoun. However, since pronouns most often refer to entities that are more central to a discourse, sentences beginning with a pronoun should often contain relevant information. In order to address this issue, we plan on incorporating an anaphora resolution module to determine the referents of at least sentence initial pronouns.

Anaphora resolution will be facilitated by the incorporation of a named entity recognizer into the summarizer. Besides being useful for anaphora resolution, a named entity recognizer should help with matching query terms against document terms. Both by providing canonical versions of names, and by providing type information about the named entities in a sentence, such a recognizer will make it possible to model the probability that a sentence is relevant to a query in a more principled way. With accurate named entity recognition and anaphora resolution, it becomes possible to calculate the centrality of a sentence, which has been shown to be a helpful feature for ranking sentences (Erkan and Radev, 2004).

In summary, this has served as a useful first step towards building a competitive summarizer, and we hope to improve it in several ways by the next DUC competition. At that time, the system will serve as a testbed for more exploratory improvements.

References

Alan Agresti. 1996. *Introduction to Categorical Data Analysis*. John Wiley and Sons, New York.

David Blei, Andrew Ng and Michael Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

Gunas Erkan and Dragomir Radev. 2004. Lexpagerank: Prestige in multi-document text summarization. In *Proceedings of EMNLP*.

Thomas Hofmann. 1999. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval*.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

Chin-Yew Lin and E.H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the Human Language Technology Conference and Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*.

Jenine Turner and Eugene Charniak. 2005. Supervised and Unsupervised Learning for Sentence Compression. In *Proceedings of the 43rd Annual Meeting of the ACL*.