Query-focused summarization by supervised sentence ranking and skewed word distributions

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

Abstract

We present a supervised sentence ranking approach for use in extractive summarization. The supervised approach achieves domain independence by making use of a range of word distribution statistics as features, of the sort typically used for unsupervised domain-independent ranking. We present empirical trials on the DUC 2006 query-directed multi-document summarization task, and demonstrate that the very general machine learning approaches taken can provide competitive results for this task. The general approach provides great flexibility for incorporating many more features.

1 Introduction

Sentence extraction summarization systems take as input a collection of sentences (one or more documents) and select some subset for output into a summary. This is best treated as a sentence ranking problem, which allows for varying thresholds to meet varying summary length requirements. Most commonly, such ranking approaches use some kind of similarity or centrality metric to rank sentences for inclusion in the summary – see, for example, Lin and Hovy (2002); Erkan and Radev (2004); Radev et al. (2004); Blair-Goldensohn (2005); Biryukov et al. (2005); Mihalcea and Tarau (2005) and the references therein. Such an approach is typically preferred over supervised ranking approaches for reasons of domain independence.

This paper presents an alternative approach, whereby a number of similarity/centrality metrics are used, not directly to rank the sentences, but rather as features within a supervised machine learning paradigm. Since the features themselves are not domain-specific, the benefit of domain generality is retained, while still accruing the benefits of supervised learning.

We examine this approach within the context of query-focused multi-document summarization, for which there is much less training data for supervised approaches than query-neutral multi-document summarization. We address this through the use of two separate ranking models: one trained on a large collection of document clusters and associated (query-neutral) manual summaries; the other trained on a smaller data set from the 2005 DUC query-focused multi-document summarization task, which includes document clusters, queries, and the associated (query-focused) manual summaries. The scores from the first ranker are used as features in the second ranker. In addition to the use of two ranking models, we achieve query responsiveness by skewing the word distributions, which make up the features of our models, towards the query. All of this is achieved within a very general supervised ranking paradigm, which is robust and domain independent.

In the next section we discuss in detail the architecture and training of our system. We broke the query-directed summarization problem down into three tasks:

1. Text normalization and sentence segmentation
2. Sentence ranking
   a. query-neutral ranking
   b. query-focused ranking
3. Sentence selection from a ranked list

We then present the results of the DUC 2006 evaluation, as well as of a number of experiments using different algorithms and features on that evaluation set.

2 Sentence extraction system

2.1 Text normalization

In the multi-document summarization data[1] made available for the Document Understanding Conferences (DUC), each document set is a collection of individual articles, each article in its own file. We created one large text file for each document set by concatenating the raw content text from each article, discarding the meta-data. We then used a simple algorithm to perform sentence segmentation, making use of a list of common abbreviations extracted from the Penn Treebank.

Feature extraction from sentences involved mapping words to a common form, by making them lower case and removing 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 do not perform any stemming or removal of stopwords at this stage. This mapping is for feature extraction only – the original sentence is maintained for eventual output by the system if selected, although some additional normalization is done at the time of sentence selection (see section 2.3).

2.2 Supervised sentence ranking

For sentence ranking, we implemented a perceptron ranker (Crammer and Singer, 2001). The perceptron ranker algorithm is an online conservative algorithm that scores examples such that each score falls into a bin, which is the rank of the example. The algorithm learns a parameter vector of weights, as well as a vector of rank boundaries. The weight vector is combined with a given feature vector via an inner-product to calculate the score of the sample. The rank boundaries are real numbers in one dimension. The ranks associated with their respective boundaries are monotonically increasing. So, a sample with rank (5) is closer to rank (6) than it is to rank (7), this is what makes it a ranking algorithm rather than a multiclass classification algorithm. The update rules for the parameter vectors change the weights such that the new score would be closer to the bin for the correct rank, and change the rank boundaries such that the boundary for the correct rank is closer to the original score. See the PRank paper (Crammer and Singer, 2001) for details.

The objective we used for our supervised ranking is the ROUGE-2 score as configured for the DUC-05 evaluation, excepting that we enabled both stemming and stop-word removal. For a 250 word summary we are typically only interested in the top 15 or so sentences in a document set (while allowing for redundancy). As a result, we configured the perceptron ranking algorithm to produce models with only 3 ranks: the top 35 sentences in each cluster, in terms of ROUGE-2 score, were in the top rank (100), the next 65 were in the middle rank (50), and the rest were in the low rank (1). Some clusters did not have 100 sentences, or had less than 100 different ROUGE-2 scores, in which case all of the lowest scoring sentences were given the low rank. Within each document cluster, feature values were normalized by dividing by the maximum absolute value of the raw feature value for any sentence in the cluster.

Unlike the perceptron ranking algorithm in Crammer and Singer (2001), we used an averaged perceptron at test time to mitigate over-fitting of the training data (Collins, 2002). This common technique averages the feature weights over all time steps when it outputs the final model, thus reducing over-fitting. At test time, we do not make use of the rank boundaries, hence calibrating the averaged parameters to the boundaries was not an issue.

Because the features that we used are both small in number and domain independent (see sections 2.2.1 and 2.2.2), there were a couple of issues related to convergence on the training data that we had to manage. First, we often had a very large number of sentences with a rank of 1 (the worst), so that the algorithm would sometimes learn to simply guess rank 1 for every sentence, since the vast majority of the sentences were of that rank. We were able to control for this in two ways: by either adding more features (e.g., unigrams); or by using a random subset of the rank 1 sentences, rather than the full set (Brinker and Hullermeier, 2005). We found that the latter technique was the better approach, since it allowed us to remain with a simple, domain independent feature set.

[1] `http://duc.nist.gov/`

A second, related issue, is that the limited feature set is such that the algorithm cannot converge to perfect ranking performance on the training set. We experimented with n-gram features, as mentioned above; although this allowed the perceptron to converge to the training data very accurately, it did not improve ranking performance against our heldout training data. We also experimented with a second order polynomial kernel for the perceptron, which increases the feature-space by effectively pairing all of the original features with one another. This also helped the perceptron to converge, but it also did not significantly help with accuracy on the heldout data.

2.2.1 Query-neutral sentence ranking

The base feature set that we use is the same as was used in our baseline system from DUC 2005 (Fisher et al., 2005). 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

$$\rho(s) = \operatorname*{average}_{z \in Z_c} (\text{rouge}(s, z))$$

where rouge(s, z) is the ROUGE score (Lin, 2004) of sentence s with z as the reference summary[2]. We calculated this value for all sentences in each cluster of the DUC 2001-2002 training data for summaries of size 200 words and 400 words, giving us our "gold standard" ranking for use in training the base system.

For each sentence in a cluster, we extracted a small number of features for ranking. 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 $c \in \mathcal{C}$. The version of TF*IDF that we make use of is:

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

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[3] is defined as follows:

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

where

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

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[4] 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 sentence we calculated 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 in the system last year was the position of the sentence in the document. All of these feature values were normalized within their document set, by dividing the raw values by the highest absolute raw value in the document set. Table 1 summarizes the base feature set.

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		

Table 1: Base feature set

Beyond these base features, we added the features from Table 1 for both the immediately previous and immediately following sentences as features for the current sentence, effectively tripling the number of features. The improvement due to these "neighbor" features is presented in section 3.

Using multiple similarity or centrality metrics as features is useful because all of these features score co-occurrence dependencies differently. For example, the log likelihood ratio captures whether a word

[2] For this work, we used ROUGE-2, with stemming and stop words removed as our score.

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

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

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, in contrast, 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, which can be quite useful for this task. Hence, including both rather than choosing between them, provides additional sensitivity to the kinds of patterns that discriminate between good and poor sentences for extraction.

To summarize, our base feature set for each sentence consists of the values of the features in Table 1 for the sentence, as well as the values for those features in the immediately previous and following sentences.

2.2.2 Query-focused sentence ranking

Lexical overlap

The baseline system from DUC 2005 (Fisher et al., 2005) achieved query-focused summaries via a crude lexical overlap metric. For a given document set, sentences were binned into three sets, depending on the number of non-stop words from the query that were in the sentence. The first set included sentences with two or more query words (not counting stop words); sentences in the second set had 1 query word; and the last set contains sentences without query words. Within each set, the sentences were ordered by the query-neutral ranking. Sentences were then selected for the summary (see section 2.3) from the first set, in order, until it was exhausted, then from the second until it was exhausted, and finally from the third set. This approach performed surprisingly well in the DUC-2005 evaluation.

One simple extension to the above approach is to allow for as many sets as there are different query word counts in sentences, rather than just 3 sets. Thus, a set for those with 10 non-stop query words, another set for those with 9, etc. In effect, this is a type of cosine similarity metric between the query and each sentence, with the query-neutral ranking being used to break ties between sentences with the same cosine score.

Skewing word distributions

To achieve query-sensitivity within the context of a single supervised ranking system, we examined skewing word distributions towards the query for purposes of calculating distribution sensitive features. Recall that we have a number of features (see table 1) that rely on the distribution of a word in the document set relative to its distribution in the corpus. We skew the word distributions towards the query in a document set by adding the counts of each of the non-stop query words, multiplied by an empirically determined factor, to the counts of words in the document set. In effect, non-stop query words have their counts increased in the document set for purposes of calculating the word-distribution sensitive features. The result is that when extracting features from a sentence, words that are in the query will have relatively larger feature values, by virtue of having higher document set counts. When the individual words have larger values, the feature values for sentences containing those words will also be higher.

Note that this approach allows us to train the models on non-skewed training data, with the query-focused skewing happening at test time. Hence, large amounts of query-neutral multi-document summarization training data can be exploited. With this approach, we can get query sensitivity within a very simple ranking approach. This has the additional benefit of being able to convert the ranking score to a normalized probability (via softmax), thus allowing the use of these scores as features in another stage of ranking.

Re-ranking

The first-pass ranking model in our approach is trained on query-neutral summarization data. Given that we now have a small amount of query-sensitive training data from the DUC-2005 evaluation set, we can build a specifically query-focused reranker from this data. As with the query-neutral ranking, we used the perceptron ranking algorithm.

The sentences are first ranked using the skewing approach described above, and the output from this step (the softmax normalized perceptron score) is one of the features input to the reranker. In addition to this feature, which has its weight empirically fixed, the reranker has two other sets of features for which it learns parameter weights. These

are features characterizing the number of non-stop query words in the sentence. We first partition the set of non-stop query words into two subsets: those with log likelihoods higher than a fixed threshold and those with log likelihoods lower than the threshold. The log likelihood is calculated for each query word for that cluster, using unskewed counts. Then, for each subset s, there are five indicator features: 0 words in the sentence from s; at least 1 word in the sentence from s; at least 2 words from s; at least 3 words; and at least 4 words. For the trials reported here, the partitioning threshold was set empirically at 10.

For training the reranker, we used 29 of the DUC-2005 document sets as training data, 5 of them as held-aside data (for stopping training), and 16 as development data for testing different features. We fixed the weight of the baseline ranker at 1000.

2.3 Sentence selection

At the sentence selection stage, we removed any sentence less than 5 words or greater than 50 words in length. The restriction on being too short is based on the intuition that in an extraction system, anything too short will be meaningless out of context. The restriction on being too long is a simple way to keep the system from extracting long lists, which generally do not make a good summary. In addition, any sentence that begins or ends with a quotation mark was also filtered out. Finally, sentences beginning with a pronoun were removed, to avoid the most obvious cases of poor anaphora resolution.

At this point we also applied some simple compression to the remaining sentences. Namely, we removed any paired parentheticals, defined as stretches of text in a sentence that were delimited by parentheses, single dashes, or em-dashes.

In the baseline system from DUC-2005 (Fisher et al., 2005), sentences were selected in order based on the final ranking, until the summary size limit was reached, with some sentences being removed for lack of novelty, as follows. Stop-words were removed from a candidate sentence, then the remaining words were stemmed and the unigram overlap with stemmed non-stop words already in the summary was calculated. If the overlap amounted to 50 percent or less of the non-stop words in the candidate, the candidate was added to the summary, otherwise it was discarded.

System	ROUGE-2	ROUGE-SU4
OGI-06	0.08525	0.14090
OGI-05	0.07601	0.13126
OGI-05+cos	0.08178	0.13764

Table 2: Comparison of our DUC-05 and DUC-06 systems tested on the DUC-06 data. The OGI-05 system was as entered at DUC-05. The OGI-05+cos system is the same system as OGI-05, but now with query-focus based on the cosine measure, rather than just the 3 bins of the original. The OGI-06 system is the system entered at DUC-06. It uses a perceptron for ranking and reranking, and has a number of other differences from the 2005 versions, as described above.

For the current system, we also select sentences in the final ranking order, while skipping some due to overlap. In this system, we removed stop words from the candidate, but did not stem the remaining words. Also, we measured bigram overlap instead of unigram overlap. In addition, rather than just using 50 percent overlap as the cutoff, we tried differing thresholds, making the threshold progressively more restrictive until the resulting ROUGE scores dropped significantly. Our final threshold was 65 percent new bigrams for a candidate to be added. Finally, we ordered the extracted sentences by document-id, and then by order they occurred in the document.

3 Results

3.1 DUC 2006 Results

The OGI-06 system was competitive in the field of participants in DUC 2006. Its ROUGE scores are shown in the first row of Table 2. According to the official results, in a field of 34 entries, the OGI-06 system was 8th in ROUGE-2, 9th in ROUGE SU-4, 4th in Basic Elements (BE) scoring, 19th in query responsiveness (manually scored), and 4th in overall responsiveness (also manual). Based on this, it appears that there is substantial room to improve our query responsiveness (see future directions), but that the summary quality that results from our general approach is relatively high. We hypothesize that the high overall responsiveness and BE scores are due to the syntactic well-formedness of the bulk of our summaries, resulting from the limited compression applied to the output.

In order to determine whether the current system

Focus Method	ROUGE-2	ROUGE-SU4
Cosine	0.08356	0.14031
Skew	0.08404	0.13768
Skew + Rerank	0.08525	0.14090

Table 3: Results from our different approaches to query-sensitivity, otherwise using the DUC-06 system. The Cosine method is a ranking by the number of query words (minus stops) in a sentence, with ties broken by the non-query-sensitive perceptron ranking. The Skew method is our approach that skews the counts of non-stop query words for purposes of calculating word statistic features (e.g., log-likelihood ratio) for ranking. The Skew+Rerank version is our submitted system as described above.

is an improvement over that of Fisher et al. (2005), we evaluated both systems against the DUC-2006 test set. We used the official DUC 2006 configurations for ROUGE. As can be seen in Table 2, the overall ROUGE scores increase substantially from the 2005 to 2006 system. Table 2 also shows that by changing the 2005 system to use the improved cosine similarity metric for query sensitivity (see discussion above) roughly half of the gap between the two systems is closed. Much of the remaining improvement is explained in the next section on the different results for query-focusing.

3.2 Query-focused sentence ranking

This year we experimented with several different methods for focusing the summaries to a query, as described above. We first tried using a cosine similarity metric between the query and candidate sentence. Then we developed a novel method that skews the word distributions towards the query before calculating word based statistical features. By skewing the word distributions we obviated the need for calculating the cosine similarity, and now had a single ranking score that proved as effective as any other method we had tried. Once we had an effective single ranking score, we used that and several other query related features in a reranker with the DUC 2005 data for training. The results are shown in Table 3 .

As can be seen in Table 3 , the cosine method and skew method perform similarly. However, the skew method allows us to incorporate its output into a reranker, which performs better than either the cosine or skew methods alone. Due to time constraints, we did not experiment with as many features in the reranker as we might have; we expect to be able to improve on that score. As mentioned previously, query responsiveness is where the system scores comparatively poorly. See section 4 for future directions.

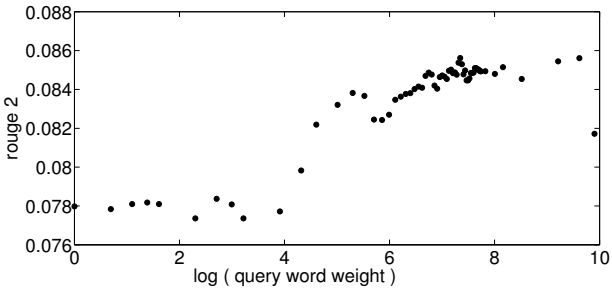

Figure 1: Natural log of the weight given to additional query word counts versus the system rouge 2 score on the DUC-2006 evaluation set.

3.2.1 Word distribution skewing

The skewing of word distributions towards the query as described above proved quite effective. The counts of each non-stop word in the query that also appeared in the document set were: (1) multiplied by a fixed weight; and (2) added to the document set counts. We determined the query word weight empirically. The additional counts applied to the document set, and thus also needed to be temporarily added to the overall corpus counts when calculating features for the document set. In Figure 1, the improvement of the system, as measured by ROUGE-2, is shown as a function of the log of the query word weighting.

For our submitted system, as well as for other feature sets we experimented with, the optimal query word weight was generally about 1000, which was the deployed value in reported systems. Based on the graph in Figure 1, the best value for this evaluation would have been at around 1500, but the sensitivity to this parameter, at least when it is above 50, is relatively low.

3.3 Feature Analysis

In order to see which features were most important for inference, we trained the system on subsets of the total feature set. A comparison of several feature configurations with the NIST baseline is given in Table 4 .

The NIST baseline is just the first n sentences up to 250 words. Our system with no features ranks

Features	ROUGE-2	ROUGE-SU4
NIST baseline	0.04947	0.09788
none	0.05547	0.10847
position	0.06650	0.11839
all	0.08404	0.13768
all + reranking	0.08525	0.14090

Table 4: Our system performance against the NIST baseline using no features, only the position feature, all features and all features plus reranking. Note that all of these systems use the query word skewing, but that it has no effect for the 'none' and 'position' models since there are no word distribution features in those models.

sentences in the same way as the NIST baseline. However, the ROUGE scores are improved over the NIST scores because our sentence selection stage filters redundant sentences. Adding only the position feature improves the performance by a large amount. The position only model simply learns to pick the first sentence in each document, but when combined with the sentence selection stage, the performance is much better than either the NIST baseline or the model with no features. The model with all features, but no reranking, is much better than the position only model. The reranking model with its current feature set is only slightly better than the skewed word-distribution only model.

3.3.1 Skewing without reranking

We evaluated the skewing without reranking configuration with a number of different feature combinations. The combination of all of our features provides the best performance on the DUC 2006 data. A summary of the system performance using different feature combinations is given in Table 5 .

It appears that the log-odds ratio may be more important than the log-likelihood ratio when skewing the counts. One reason for this may be that the log-odds ratio can be negative, indicating an anti-correlation, while the log-likelihood ratio is always positive, indicating the degree of surprise. When we skew the distributions of the query words, words in the document set that are overall very common (e.g., *president*) may now become very uncharacteristic of the document set. This actually increases the log likelihood ratio score for that word, since a high score indicates either surprisingly many or surprisingly few co-occurrences. The log odds score avoids this ambiguity, since surprisingly few co-occurrences are scored negatively.

Features	ROUGE-2	ROUGE-SU4
position	0.06650	0.11839
tfidf+position	0.06467	0.11707
ll+position	0.07418	0.12602
lo+position	0.07535	0.12665
tfidf+ll+position	0.07981	0.13561
tfidf+lo+position	0.08264	0.13425
ll+lo+position	0.07744	0.12982
tfidf+ll+lo+position	0.08404	0.13768

Table 5: The different features are: sentence position, TFIDF, log-likelihood ratio, log-odds ratio. Each row shows the set of features included in the model. All of these models used word distribution skewing without reranking.

Because the denominator of our tf.idf score is the raw inverse document frequency, it is particularly high for words that occur in no other cluster, relative to the log-likelihood and log-odds. This favors very specific terms, for example, proper nouns over common nouns, even those that may be highly correlated with the cluster. Hence, while this feature does not perform particularly well on its own – as shown in Table 5 – it is very complementary with the other scores, and provides substantial system improvements when included with the other scores. When used with the log likelihood or log odds scores, the ranker learns negative weights for the tf.idf features, thus penalizing words when they have high tf.idf feature values. If those words do not have high log likelihood or log odds scores to offset such a penalty, they will be dispreferred.

3.3.2 Reranking

Evaluating different feature configurations of the base ranker when using the reranker did not provide as interesting results as with the skewing models. This is because the reranker features are themselves log-likelihood based, via the partitioning threshold, and thus even when the log-likelihood features are removed from the base ranker, something similar is still being used by the reranker. The small difference between just the position feature and all features shown in Table 6 is the result of the reranker having extra log-likelihood features that are always being used.

In order to determine the utility of the surrounding sentence features , we ran the evaluation with and without those features. We left all of the base features in, but removed those features for either the

Features	ROUGE-2	ROUGE-SU4
position	0.08355	0.13930
all features	0.08525	0.14090

Table 6: Performance of the reranker system when the base ranker uses just the position feature, or all of the features.

previous, the following, or both the previous and following sentences. These configurations, summarized in Table 7, are with reranking. Not surprisingly, having the statistics for either the previous sentence or both of the surrounding sentences improves the overall score. Having features for the statistics of just the following sentence does not help, if anything it hurts performance by a little.

4 Summary and future directions

Our new summarizer substantially improves over our 2005 entry at DUC. We improved query-sensitivity by introducing a novel approach that skews word distributions in a document set towards the query, while using a query-neutral ranker with those skewed distributions. The new summarizer improved performance further by making use of a query-focused reranker that takes as input the skewed query-neutral rankings, as well as some query related features. Using this query-focused reranker was possible due to the DUC-2005 data for training the model. In addition, we improved through the use of novel features, such as the "neighboring" sentence features.

There are a number of ways to improve the current system. The current system does not perform any query expansion. Query expansion should work well with our word distribution approach, and it appears to help other systems significantly. The feature set for the reranker is another area we will explore, as we experimented with relatively few different features for the current system. Though including all unigrams as features led to over-fitting, we would like to find a subset of lexical n-gram features that are relevant to indicating importance and applicability to inclusion in a summary. We also want to include features that are indicative of what sort of question the query is. Another set of features to explore are discourse connectives, and how they relate one clause to another. Because of the general machine learning framework, incorporation of a range of additional features (e.g., query expansion or discourse segmentation) or stages of processing (e.g., anaphora resolution) is relatively straightforward.

Features	ROUGE-2	ROUGE-SU4
neither	0.08243	0.13826
previous	0.08367	0.13952
following	0.08181	0.13878
both	0.08525	0.14090

Table 7: The change in score as a result of surrounding sentence features. These results are for the system using the reranker.

References

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

M. Biryukov, R. Angheluta, and M.F. Moens. 2005. Multidocument question answering text summarization using topic signatures. *Journal on Digital Information Management*.

S. Blair-Goldensohn. 2005. Columbia University at DUC 2005. In *Document Understanding Workshop (DUC) 2005*.

K. Brinker and E. Hullermeier. 2005. Calibrated label-ranking. In *Learning To Rank Workshop, NIPS 2005*.

M.J. Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–8.

K. Crammer and Y. Singer. 2001. Pranking with ranking. In *Neural Information Processing Systems*. NIPS.

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

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

S. Fisher, B. Roark, J. Yang, and B. Hersh. 2005. OGI/OHSU baseline query-directed multi-document summarization system for duc-2005. In *Proceedings of the Document Understanding Workshop (DUC)*.

C.Y. Lin and E. Hovy. 2002. Automated multi-document summarization in NeATS. In *Proceedings of the Human Language Technology Conference*.

C.Y. Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *Workshop in Text Summarization, ACL'04*.

R. Mihalcea and P. Tarau. 2005. An algorithm for language independent single and multiple document summarization. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*.

D. Radev, T. Allison, S. Blair-Goldensohn, J. Blitzer, A. Çelebi, S. Dimitrov, E. Drabek, A. Hakim, W. Lam, D. Liu, J. Otterbacher, H. Qi, H. Saggion, S. Teufel, M. Topper, A. Winkel, and Z. Zhang. 2004. MEAD - a platform for multidocument multilingual text summarization. In *LREC*, Lisbon, Portugal.