OHSU Summarization and Entity Linking Systems

Seeger Fisher, Aaron Dunlop, Brian Roark, Yongshun Chen, Joshua Burmeister
Center for Spoken Language Understanding
Division of Biomedical Computer Science
Oregon Health & Science University
{fishers,dunlopa,roark,chenyo}@cslu.ogi.edu, burm0248@gmail.com

Abstract

We present two distinct text analysis systems.

We first present two supervised sentence ranking approaches for use in extractive update summarization. For the first, we use the same general machine learning approach described in Fisher and Roark (2008) for update summarization. In the second, we use a similar machine learning approach, but include sub-sentential units produced by our discourse segmenter, see Fisher and Roark (2007b), as possible units for inclusion in a summary. Interestingly, we find that one approach performs significantly better in the production of the base summary, while the other approach performs significantly better in the update summary.

We then present a large-corpus entity linking system. This system expands queries using internal links within Wikipedia and link entities with minimum-spanning-tree clustering. We present and evaluate empirical results on the TAC 2009 knowledge-base-population data, and demonstrate competitive results with a simple system.

1 Introduction to Summarization

Our approach to the update summarization task is similar to our approach to the query-focused multi-document summarization task, with some changes to how we filter redundant sentences within a summary. Furthermore we extend our approach to not only include full sentences from the source documents in our summaries, but also sub-sentential units called "elementary discourse units" or *edus*. These *edus* are usually a single clause but can sometimes be smaller. Our summarization approach is a form of extractive summarization. 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.

We present 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 and 2006 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.

We broke the query-directed summarization problem down into several tasks:

1. Text normalization
2. Segmentation
   a. document to sentences
   b. sentence to *edus*
2. Sentence ranking
   a. query-neutral ranking
   b. query-focused ranking
3. Sentence selection from a ranked list

In previous papers we have detailed the architecture and training of our query-focused multi-document summarization system (Fisher and Roark, 2006; Fisher and Roark, 2007a; Fisher and Roark, 2008). In this paper we review the previous summary system, and then show how we modified our approach to produce update summaries both with only sentences as the units being extracted and with either sentences or *edus* as the units being extracted.

1.1 Sentence Extraction System

Most stages of our sentence extraction system are detailed in Fisher and Roark (2006). We give just a brief review of the stages here. The exception is 2b, segmentation of sentences into *edus*, which we cover in more detail.

1.2 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.

1.3 Segmenting sentences to *edus*

One failing of extractive summarization at the sentence level is that sentences often are quite long and can contain substantial amounts of information that is not very important. This drawback is even more pronounced when producing query-focused extractive summaries. In order to circumvent this issue, this year we experimented with extracting sub-sentential spans of text. Rather than simply segmenting each sentence into its constituent clauses, we instead segment into *edus*, which form more natural units of meaning in discourse. We then allowed the system to rank whole sentences and *edus* together as if they were all sentences in the original system.

We use the system fully described in Fisher and Roark (2007b) to perform segmentation of individual sentences into its *edus*. A brief summary of that approach follows. We use a log-linear model to individually classify each word in a sentence as to whether it is the end of a segment or not. We make a zero-order Markov assumption, that is, each word is classified independently of the classification results of its two neighbors.

There are several classes of features the classifier uses. N-gram features for the lexical items, part-of-speech tags, and shallow phrase tags are used. We use up to 6-grams, with a max of 3 items to either side of the possible segmentation boundary we are currently predicting, without any skips. We also use features derived from full syntactic parses. These features are all derived from the lowest subtree of the full parse of the sentence that spans the possible segmentation boundary that we are predicting. The features are both lexicalized and unlexicalized versions of productions and parts of productions extracted from that subtree of the syntactic parse. See Fisher and Roark (2007b) for the exact details.

1.4 Supervised sentence ranking

For sentence ranking, we implemented a perceptron ranker (Crammer and Singer, 2001). The objective we used for our supervised ranking is the ROUGE-2 score as configured for the DUC-06 evaluation. 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. Within each document cluster, feature values were normalized.

Using a limited feature set, the algorithm can-

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

1.	average tf.idf	6.	average LO
2.	sum tf.idf	7.	sum LO
3.	average LL	8.	sum (max 3) LO
4.	sum LL	9.	Sentence position
5.	sum (max 3) LL	10.	centrality

Table 1: Base feature set. LL is log-likelihood, and LO is log-odds

not converge to perfect ranking performance on the training set. We experimented with n-gram features, but although this allowed the perceptron to converge to the training data very accurately, it did not improve ranking performance against our held-out training data. We also experimented with a second order polynomial kernel for the perceptron. This also helped the perceptron to converge, but it did not significantly help with accuracy on the heldout data. See Fisher and Roark (2006) for further details.

1.4.1 Query-neutral sentence ranking

The base feature set that we use is the same as was used in previous systems (Fisher and Roark, 2007a; Fisher and Roark, 2008). 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

$$\rho(s) \quad = \quad \underset{z \in Z_c}{\text{average}} \left(\text{rouge}(s, z) \right)$$

where rouge(s, z) is the ROUGE score (Lin, 2004) of sentence s with z as the reference summary. We calculated this value for all sentences in each cluster of the DUC 2001-2003 training data for summaries of size 100, 200 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. The feature set is summarized in Table 1. See Fisher and Roark (2006) for details on calculation of the features.

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.

Using multiple similarity metrics as features is useful because all of these features score co-occurrence dependencies differently.

1.4.2 Query-focused sentence ranking

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 query-sensitive training data from the DUC-2005 and 2006 evaluation set, we can build a specifically query-focused reranker from this data. As with the query-neutral ranking, we used the per-

ceptron 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. See Fisher and Roark (2007a) for further details on this approach.

For training the reranker, we used the DUC-2005 document sets as training data, and the DUC-2006 document sets as development data for testing different features. We fixed the weight of the baseline ranker at 1000.

Query expansion

Besides skewing word distributions towards the query, and then performing re-ranking with query-based features, we also performed query expansion to make our sentence ranking more sensitive to the query. We used a 300 million word corpus to estimate the probability that two words will occur in adjacent sentences. We picked the 100 non-stop words with the highest log-likelihood as expansion terms for each query term. These expansion terms were included in the re-ranking step described above, but as separate features from the actual query terms. For more details see Fisher and Roark (2007a).

1.5 Sentence and *edu* selection

At the sentence selection stage, we removed any sentence (or *edu* 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.

For the base summary in a docset, sentences (and *edus* for our second system) were selected in order based on the final ranking, until the summary size limit was reached, with some sentences (and *edus*) being removed for lack of novelty, as follows. Stopwords were removed from a candidate span, then the bigram overlap with non-stop words already in the summary was calculated. If the overlap amounted to 65 percent or less of the non-stop words in the candidate (determined empirically), the candidate was added to the summary, otherwise it was discarded. Finally, we ordered the extracted spans by document-id, and then by order they occurred in the document.

2 Update Summaries

Our system for producing the update summary, the second summary for each docset, is very similar to the query-focused multi-document summarizer just described, but with some important differences in selection from the ranked list of sentences (and *edus* in the second system). We use the same classifier and feature set, trained in the same way as the base summarizer. However, in the sentence selection stage when checking for overlap between a candidate sentence or *edu* and the spans already in the summary so far, we checked not only against spans already in the new summary, but also against spans from the summary of the first partition. Thus, there was no change to our ranking algorithm, only to the part of the system that adds already ranked spans to the growing summary.

3 Summarization Results

This year at TAC, each team was allowed to submit two runs, each of which was fully evaluated, both automatically and manually. Our first run, CSLU-OHSU1, used precisely the same model (no retraining) as our 2008 system (Fisher and Roark, 2008). Our second run, CSLU.OHSU2, was trained on the same training data as the first, but with the inclusion of the sub-sentential units, *edus*, described above in section 1.3.

The difference between the two runs was somewhat surprising. We will concentrate here on the manual pyramid evaluations (recall based), as they are most indicative of the quality of the content in the summary, which is what we were aiming to maximize. The CSLU.OHSU1 run scored at about the top of the bottom third (35th of 55) in the base summary, which is a little worse than last year, and indicates that as in previous years, participants are getting better (this summarizer scored in the top quartile for virtually all measures in the 2006 evaluation). We were not surprised that the second run, CSLU.OHSU2, score significantly better for the base summary, about the bottom of the top third (19th of 55). Because the second run could choose from smaller units, a larger number of relevant units should be included in the summary - and the measures we are looking at are recall based.

The surprise came when we examined the results of the update summaries. For the update summaries, the first run improved quite a bit, and the second run fell dramatically. We expected that even if our first run improved, the second run would still be better. Instead, our second run (CSLU.OHSU2) fell to 38th of 55, while the first run improved to 20th of 55. The improvement of our first run can be attributed to our approach to creating the update summary, and agrees with results from last year in which our update summaries scored better than our base summaries. The fall in score for the second run was puzzling at first. What appears to have happened is a combination of two factors. First, our sentence selection filter for update summaries (see section 2) became too restrictive for the shorter units (*edus*) which were now being included in both the base and update summaries, i.e. we did not re-estimate the filtering threshold for our second run, and we should have. Second, even if the threshold is changed, there is still a problem. That is that when an important sentence is segmented, the query-sensitivity as we measure it (akin to word overlap, where word can also be a synonym, with the query) may leave one of the segments as scoring with little or no overlap, even though it contains relevant information. The solution to this problem is not readily apparent, but we have a number of ideas. One idea is to include the score of other *edus* in the same sentence as features, similarly as to how we include the scores of neighbor sentences as features in the base summarizer. Another idea is to use the rhetorical relationships between *edus* in the same sentence as features, thus allowing the discourse structure to play a role in scoring sub-sentential spans. We currently don't have results from using these ideas, but are planning to implement them.

4 Summary and future directions

We have presented the application of general supervised machine learning techniques to the problem of sentence ranking for extractive summarization. By exploiting model summaries to define a gold-standard ranking over sentences, we can use well-motivated learning approaches, which handle an arbitrary number of features. We have demonstrated that many common metrics used for sentence ranking can be combined into a single ranking model that provides better performance than any of the metrics in isolation. We straightforwardly extended the model to include features of neighboring sentences, which was demonstrated to improve performance. We have applied this approach to query-directed update summarization through a number of techniques: (1) query word count inflation; (2) reranking based on query-directed training data; and (3) query expansion techniques. The resulting approach is competitive, and its generality and ease of extension should allow for substantial future developments.

There are a number of ways to improve the current system. The feature set for the reranker is an area we will continue to explore, since we have 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.

Given the intriguing results from using subsentential RST segments rather than just sentences alone as the spans extracted, we would like to experiment with going one step further and performing full Rhetorical Structure Theory discourse parses of the source documents. Previous work has shown that such parses can be useful for producing query-neutral summaries. We feel it is highly likely that they can be useful in query-focused summarization as well.

5 Introduction to Entity Linking

We now move on to describing an entity linking system for knowledge base population.

Many textual information extraction pipelines begin by assigning links between texts (documents, paragraphs, sentences, or fragments) which refer to the same entity. Indexing and search are fairly well understood, so retrieving information about an entity is usually easy, but identification of the desired information within a set of search results can be quite difficult. For example, if we wish to extract facts about the Arizona Cardinals football team from a set of search results, we might need to identify (and exclude) extraneous documents about birds, about the St. Louis Cardinals baseball team, and perhaps even some about Catholic church officials. An entity linking system assigns links between texts or from texts to previously known entities, enabling accurate information extraction.

We developed the system described here to compete in the 2009 TAC Knowledege Base Population (KBP) task (NIST, 2009). The KBP task included two subtasks, the first of which was intended to encourage and measure progress on entity-linking.

The KBP dataset included a sizable corpus of newswire text (source documents) and a partially pre-populated knowledge-base (KB). KB entries were divided into three categories:

- PER: Person
- GPE: Geo-political Entity (e.g. city or country)
- ORG: Organization (e.g. company or government office)

Each KB entry included an entity ID, a canonical name and a disambiguating document describing the entity; some included a few pre-populated facts as well. Each entity-linking query specified the name of the desired entity (which often differed from the KB canonical name of the intended entity) and referenced a source document which mentioned the entity by name. The system responds with either an entity ID or *NIL* if no link could be found.

6 System Description

6.1 Text Preprocessing

We split the source documents into paragraphs using the SGML paragraph markers, and into sentences using LingPipe's rule-based sentence segmenter (Alias-i, 2009). We removed and ignored all other SGML markup. We stored each sentence from the approximately 1.2 million source documents in a relational database, indexed by document, sentence, and paragraph. We indexed the full text of each sentence using Apache Lucene (Apache, 2009).

We stored the entities, facts, and disambiguating documents from the knowledge-base in the same database, again using LingPipe to sentence-segment the source documents. We compressed all text stored in the database, resulting in a sizable but manageable database of approximately 12 GB including all indices.

The relational database allows easy retrieval of any subsection of a document. In particular, we're able to easily and efficiently retrieve any sentence or series of sentences, allowing us to vary the size of the clustering context (see section 6.3) by parameterizing the SQL select.

We counted word occurrences in the set of source documents $\mathbb{S}$, and define $o(w)$ as the count of documents from $\mathbb{S}$ in which word w occurs and

$$idf(w) = log_e \left(\frac{|\mathbb{S}|}{1 + o(w)} \right)$$

6.2 Entity Lookup and Query Expansion

We want the ability to link a query to an entity whose name is not an exact (or even approximate) textual match. For instance, we would like the possibility of linking the query 'GOP' to the entity 'Republican Party', even though no reasonable lexical metric would associate the two. In our context, this

is a query-expansion task — given the query term 'GOP', we want to execute searches on our source documents and knowledge base for the term 'Republican Party' as well as for 'GOP'.

Various methods of automatically learning such associations have been explored, but we elected to make use of easily-available human annotations for this task. Wikipedia articles link extensively to other articles within Wikipedia, providing a rich source of potential 'aliases' for known entities. We used a November 2008 snapshot of Wikipedia, made available by New York University (Sekine, 2009) to extract the text of internal links within Wikipedia to the articles in the reference knowledge base. We used the title of each link as an alias for the subject of the target article. For example, the article on Bill Clinton is linked from elsewhere in Wikipedia as 'William Clinton', 'Clinton, Bill', 'I never inhaled', etc. These links provide a noisy but useful source of query term expansions. We stored these aliases in the database as well, referring relationally the target entities.

With each query, we construct a set of *contexts* $\mathbb{C} = \{c_1, c_2, \ldots, c_n\}$, in which each c_i is a string of tokens, generally one or more sentences, and cluster $\mathbb{C}$.

We first attempt to match the query directly to an entity name. We require an exact prefix match of 3 characters, and a total similarity of more than 80% of the query length, using a character-wise Levenshtein distance similarity. We add the entire disambiguating document for any matching KB entity to $\mathbb{C}$.

We then attempt to match the query to any aliases, applying the same 3-character / 80% threshold, and again adding the disambiguating KB documents of any referenced entities to $\mathbb{C}$. If we are unable to match any knowledge-base entities by either name or alias, we output *NIL*.

Having found one or more knowledge-base entities, we query our Lucene text index and add to $\mathbb{C}$ any sentences from the source document set which match the query, again requiring a 3-character prefix match, but this time with a 90% similarity threshold. Note that we do not attempt to match source document sentences by alias.

Finally, we add the entire disambiguating document referenced by the query.

So, $\mathbb{C}$ includes:

- The entire document referenced in the query itself.
- Disambiguating documents describing all KB entities matched either by name or by alias.
- All source document sentences which closely match the query.

6.3 Clustering

We use a simple weighted variation of the standard TF-IDF similarity metric (Robertson and Sparck Jones, 1976) to cluster related texts in $\mathbb{C}$. For any string c_i from $\mathbb{C}$, we define:

$$w_1 \ldots w_n = \textit{the set of unique words in } c_i$$
$$cnt(c_i, w_j) = \textit{occurrences of word } w_j \textit{ in } c_i$$
$$weight(c, w) = idf(w) \cdot cnt(c, w)$$

And finally,

$$distance(c_1, c_2) = 1 - \frac{\sum_{j=1}^{n} \left(idf(w_j) \sqrt{cnt(c_1, w_j) \cdot cnt(c_2, w_j)} \right)}{\sqrt{\sum_{j=1}^{n} weight(c_1, w_j) \cdot \sum_{j=1}^{n} weight(c_2, w_j)}}$$

To cluster, we apply Kruskal's minimum-spanning-tree algorithm, as described in CLR (Cormen et al., 2009) and implemented by LingPipe (Alias-i, 2009). We let e be the number of entities found through query expansion, and split $\mathbb{C}$ into $min(2e, \frac{|\mathbb{C}|}{2})$ clusters. We then link the query to the knowledge-base entity nearest to the query document and within the same cluster. If no KB entity is clustered with the query document, we return *NIL*.

7 Entity Linking Results

The TAC 2009 KBP query set consisted of 3904 queries, of which we correctly identified 2572. We present our performance as compared to the median of all systems evaluated:

Queries	OHSU	Median
All	**0.6588**	**0.7108**
non-NIL	0.4854	.6352
NIL	0.7891	0.7891

Our overall performance falls close to the median of all submissions; in fact, our performance on queries for entities which are not in the KB (*NIL*) is precisely the median. However, we significantly underperform the median system(s) on non-NIL entities. When considering potential improvements, we further subdivide our 1332 errors as follows:

- A - Failed to link (chose *NIL* when the entity was in the KB): 492
- B - Linked to the incorrect entity: 370
- C - Linked when the entity was not in the KB: 470

Classes A and B both cause errors on non-NIL entities, but we believe the causes and potential solutions are different enough to warrant considering them separately. In general, class A errors are caused by insufficient query and alias resolution. We found that 483 of the 492 Class A errors occurred when we were unable to find a single entity to cluster ($e = 0$). Thus, only 9 of the class A errors can be attributed to clustering failure, whereas all class B and C errors are clearly caused by clustering.

Note that our errors are distributed fairly evenly across those three categories. We believe this indicates a system with fairly balanced failure modes; unfortunately, it also means that a changing the operating point with a simple parameter change is unlikely to improve overall performance significantly.

The full system ran in approximately 1.5 hours of wall-clock time, consuming 13.2 CPU-hours on a heterogenous 30-node cluster.

8 Future Directions

Within the current architecture of our system, we see two primary avenues for improvement, as well as a number of parameter tuning opportunities. As noted, our errors are fairly well-balanced. We believe that errors of class A are best addressed by expanding and improving query expansion and that errors of classes B and C call for improved clustering.

8.1 Query Expansion

We believe our query expansion would be improved by weighting a proposed alias match by the number of links within Wikipedia using the alias. We plan to try weighting proposed entities (whether matched by the exact query or by alias) by the number of links incoming to that entity. We would first explore a simple weighting by count, followed perhaps by PageRank over all internal Wikipedia links. However, note that expanded QE could potentially shift errors (from class A to other classes) instead of eliminating them; improvements in clustering are needed as well.

8.2 Clustering

Our simple TF/IDF context distance metric (see section 6.3) is surprisingly effective, but clustering could probably be improved by substituting a more principled metric. In particular, we believe we could improve system performance by adding a named entity recognition (NER) system within the distance metric. When comparing and linking contexts to one another, infrequent words (IDF) are of interest, but entity names probably more so. We could of course emphasize any approximate matches to the query, but we're presumably already accounting for those similarities in our choice of contexts to cluster. Additionally, we would like to benefit from common mentions of other entities. For instance, two contexts which each contain the name of a PER query's spouse, or two which contain the name of the mayor of a city (GPE) should logically be closely related. We believe incorporating NER into our distance metric would improve recognition of these similarities.

We intend to experiment with the lengths of the contexts added to $\mathbb{C}$. Some of the contexts are document-length (for example, the query document), while others are of sentence length. Although our database implementation allows fairly transparent variation of the context size, we have not yet explored the effects of such variation. We expect the clustering to perform better with contexts of similar length, and we will attempt to verify this empirically.

We plan to explore more thoroughly the performance effects of the various parameterizations described in sections 6.2 and 6.3. In particular, we chose the 3-character prefix primarily for efficiency reasons, and it occasionally misses correct matches. We chose the 80% and 90% thresholds (see section 6.2) and the cluster count arbitrarily, and intend to further explore the system's sensitivity to changes in

these parameters.

8.3 Slot Filling

Finally, we plan to complete a slot-filling system based on entity links produced by our existing system (see (NIST, 2009)). We developed some slot-filling components, but did not have a full system ready for the 2009 KBP competition.

9 Entity Linking Summary

We presented a simple entity-linking system using internal Wikipedia links for query expansion. We evaluated on the TAC 2009 KBP dataset and demonstrated reasonable performance and efficient runtime behavior. We discussed various avenues of exploration for future improvements.

References

Alias-i. 2009. LingPipe 3.8.1. http://alias-i.com/lingpipe.

Apache. 2009. Lucene. http://lucene.apache.org/.

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*.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms, Third Edition*. The MIT Press, Cambridge, MA, 3rd edition.

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

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

S. Fisher and B. Roark. 2006. Query-focused summarization by supervised sentence ranking and skewed word distributions. In *Proceedings of the Document Understanding Workshop (DUC)*.

S. Fisher and B. Roark. 2007a. Feature expansion for query-focused supervised sentence ranking. In *Proceedings of the Document Understanding Workshop (DUC)*.

S. Fisher and B. Roark. 2007b. The utility of parse-derived features for automatic discourse segmentation. In *Proceedings of the 45th Annual Meeting of ACL*, Prague, Czech Republic.

Seeger Fisher and Brian Roark. 2008. Query-focused supervised sentence ranking for update summaries. In *Document Understanding Workshop*.

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)*.

NIST. 2009. Task description for Knowledge-Base polulation at TAC 2009.

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.

S. E. Robertson and Karen Sparck Jones. 1976. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146.

Javier Artiles Satoshi Sekine. 2009. Tagged and cleaned wikipedia. http://nlp.cs.nyu.edu/wikipedia-data/.