Thomson Reuters at TAC 2008: Aggressive Filtering with FastSum for Update and Opinion Summarization

Frank Schilder Ravi Kondadadi Jochen L. Leidner Jack G. Conrad
Thomson Reuters Corporation
Research & Development
610 Opperman Drive, Saint Paul, MN 55123 USA
FirstName.LastName@ThomsonReuters.com

Abstract

In TAC 2008 we participated in the main task (Update Summarization) as well as the Sentiment Summarization pilot task. We modified the FastSum system (Schilder and Kondadadi, 2008) and added more aggressive filtering in order to adapt the system to update summarization and sentiment summarization. For the Update Summarization task, we show that a classifier that identifies sentences that are similar to typical first sentences of a news article improves the overall linguistic quality of the generated summaries. For the Sentiment Summarization pilot task, we use a simple sentiment classifier based on a gazetteer of positive and negative sentiment words derived from the General Inquirer and other sources to produce opinion-based summaries for a collection of blog posts given a set of positive and negative questions.

1 Introduction

Automatically produced summaries that do not take into account what the reader already knows arguably waste his or her time. For the TAC 2008 *Update Task*, systems were expected to produce summaries that represent a gist of what the user has *not seen* before. To this end, we employed a modified version of FastSum (Schilder and Kondadadi, 2008), a fast query-based multi-document summarizer based solely on word-frequency features over clusters, documents and topics. Summary sentences are ranked by a regression Support Vector Machine (SVM). We extended FastSum with a set of new features and filters described below.

Our submission for the update summarization task focused in particular on improving the linguistic quality of the summaries generated by our FastSum system. FastSum generates multi-document summaries without requiring computationally intensive NLP techniques such as parsing. Experiments on last years' multi-document summarization tasks showed that our system obtains ROUGE scores comparable with the top-performing systems. However, the simple word-based features (e.g., cluster/document frequency) cannot ensure a high linguistic quality in terms of coherence, focus and referential clarity of the generated summaries. In order to improve the linguistic quality of the summaries, we developed a *first sentence classifier* trained on news messages from the AQUAINT-2 corpus, because we noticed that first sentences in news messages are very focused and rarely contain anaphoric expressions to pass. This classifier can extract sentences from the entire article that resemble first sentences in news articles. We use this classifier as a filter before we apply FastSum on the remaining sentences. We also added supplemental features for the actual summarization task (e.g., ratio between the number of old versus new entities that are mentioned in a sentence).

We selected an optimal set of features according to a feature selection algorithm called LARS (Efron et al., 2004) for one of our runs. The automatic and manual scores showed the run that utilized the full set of features (20 features) to be often only marginally better than the run that only used the optimal set (9 features).

A simple baseline was submitted as our third run. This run was not manually evaluated, but received relatively high ROUGE scores. This baseline selects the temporally ordered first sentences from each article until the word limit is reached. It also performs

cosine similarity-based redundancy removal.

A need for concise summaries does not stop at facts. Increasingly, monitoring *sentiment* plays a role in business and public life (*who* thinks *what* about *whom*). We describe how our summarization method can be extended to produce opinion-based summaries for the *Sentiment Summarization Pilot Task*. As another modification to FastSum, we use a simple sentiment classifier based on a gazetteer of positive and negative sentiment words derived from the General Inquirer and other sources. Only sentences that show the same sentiment as the question are selected for a summary. Because no task-specific training data for this task was available. The weights for the FastSum features for this task were derived from training on news data.

The two runs we submitted differed in the type of gazetteers we used. The first run used a collection of sentiment words collected from the General Inquirer and other sentiment-related sources. The second run utilized a pruned and human-reviewed gazetteer.

The rest of the paper is organized as follows. Section 2 provides information on the exact task definitions and the original FastSum system. Section 3 describes our approach to the Update Summarization task while Section 4 provides the same for the Opinion Summarization task. In Section 5, we present our conclusions, address the activities as a whole, and discuss future work.

2 Background

This section briefly describes the tasks we participated in this year as well as our base system called FastSum (Schilder and Kondadadi, 2008).

2.1 Task descriptions

Main task: update summarization This year's main task addressed the challenge of providing an update summary for a cluster of documents, given that the user has already read documents on this topic. Consequently, the update summary should *not* contain information that the user is already aware of.

More precisely, the task is divided into two subtasks. The goal of the first summarization sub-task is to produce a normal query-based multi-document summary of a cluster of news documents. The second sub-task assumes that the information described in the first cluster is already known to a user who would like to receive a summary for a second cluster.

The input for this entire update task is a list of *topics*, each of which contain a title, a sequence of questions and two clusters of 10 documents each. Figure 1 shows the title and questions for one sample topic. The first cluster of documents needs to be summarized as a multi-document summary, whereas the second cluster is to be summarized taking into account the knowledge present in the information described by the first cluster.[1]

```
<title>
Kyoto Protocol Implementation
</title>
<narrative>
Track the implementation of key
elements of the Kyoto Protocol by
the 30 signatory countries.
Describe what specific measures
will actually be taken or not taken
by various countries in response
to the key climate change mitigation
elements of the Kyoto Protocol.
</narrative>
```

Figure 1: An update summarization topic

Pilot task: sentiment summarization The Sentiment Summarization Pilot Task was concerned with summarizing blog entries with respect to sentiment-related questions. The goal was to generate well-organized, fluent summaries of opinions about specified expressions ("targets"). Figure 2 contains two questions about *Rudy Guiliani*.

```
<target id = "9901" text = "Rudy
Guiliani presidential chances">
<q id = "9901.1" type= "SquishyList">
What did American voters admire
about Rudy Guiliani?
</q>
<q id = "9901.2" type= "SquishyList">
What qualities did not endear Rudy
Guiliani to some American voters?
</q>
```

Figure 2: Sentiment questions

The new challenges of this task introduced sentiment analysis and the less refined nature of the (blog) data, which was not as coherent and well-edited as the news data used in previous years within the Document Understanding Conference (DUC).

[1] This summarization scenario is especially relevant in the event of disasters, when breaking news unfolds.

2.2 FastSum system description

FastSum is a multi-document summarization system that uses a regression SVM for training a sentence classifier for good summary sentences similar to (Li et al., 2007). A part of FastSum is a filtering component that sorts out sentences that are unlikely to be in a good summary (e.g., no word overlap between query and sentence, difference in length).

Pre-processing and filtering The pre-processing module carries out tokenization and sentence splitting. In addition, a sentence simplification component based on a few regular expressions removes unimportant components of a sentence (e.g., *As a matter of fact,*). This processing step does not involve any syntactic parsing. As an initial filter, we ignore all sentences that do not have at least two exact word matches or at least three fuzzy matches with the topic description.[2]

Feature set Features are mainly based on frequencies of words in sentences, clusters, documents and topics. The features we used can be divided into two sets: word-based and sentence-based. Word-based features are computed based on the relative frequency of words for different segments (i.e., cluster, document, topic title and description). At runtime, the different relative frequencies of all words in a candidate sentence, s, are added up and normalized by the length $|s|$. Sentence-based features include the length and position of the sentence in the document.

Topic title frequency: the relative topic title word frequency for a title $\mathcal{T}$ given a sentence s: $\frac{\sum_{i=1}^{|s|} f_{\mathcal{T}}(t_i)}{|s|}$,
where $f_{\mathcal{T}} = \begin{cases} 1 & : \quad t_i \in \mathcal{T} \\ 0 & : \quad otherwise \end{cases}$

Topic description frequency: the relative topic description word frequency for a description $\mathcal{D}$ given a sentence s: $\frac{\sum_{i=1}^{|s|} f_{\mathcal{D}}(t_i)}{|s|}$,
where $f_{\mathcal{D}} = \begin{cases} 1 & : \quad t_i \in \mathcal{D} \\ 0 & : \quad otherwise \end{cases}$

Content word frequency: the relative content word frequency $p_c(t_i)$ of all content words $t_{1..|s|}$ occurring in a sentence s. The content word probability is defined as $p_c(t_i) = \frac{n}{N}$, where n is the number of times the word occurred in the cluster and N is the total number of words in the cluster: $\frac{\sum_{i=1}^{|s|} p_c(t_i)}{|s|}$

Document frequency: the relative document frequency $p_d(t_i)$ of all content words $t_{1..|s|}$ occurring in a sentence s. The document probability is defined as $p_d(t_i) = \frac{d}{D}$, where d is the number of *documents* the word t_i occurred in for a given cluster and D is the total number of documents in the cluster: $\frac{\sum_{i=1}^{|s|} p_d(t_i)}{|s|}$

Headline frequency: the relative headline word frequency of all content words in a sentence s. The headline probability is defined as $p_h(t_i) = \frac{h}{H}$ where h is the number of times the word occurred in the headline and H is the total number of words in the headline: $\frac{\sum_{i=1}^{|s|} p_h(t_i)}{|s|}$

Sentence length: a binary feature with a value of 1 if the number of words is between 8 and 50 and zero otherwise.

Sentence position (binary): indicates whether the position of the sentence is less than a predefined threshold.

Sentence position (real): the ratio of the sentence position over the number of sentences in the document.

Training In order to learn the feature weights, we trained a regression SVM (Joachims, 2002) on the previous year's news data using the same feature set. In regression, the task is to estimate the functional dependence of a dependent variable on a set of independent variables. In our case, the goal is to estimate the "summary-worthiness" of a sentence based on the given feature set. In order to get training data, we computed the word overlap between the sentences from the document clusters and the sentences in DUC model summaries. We associated the word overlap score to the corresponding sentence to generate the regression data. Note that this is the overlap score based on exact matches, and

[2]Fuzzy matches are defined by the OVERLAP similarity (Bollegala et al., 2007) of at least 0.1.

not the OVERLAP score used for computing fuzzy matches, as described in the previous section.

3 Update summarization

For the update task we made two changes to our FastSum system: we added more features that would penalize sentences that are similar to the ones from the previous cluster. To improve on the linguistic quality, we also added a first-sentence classifier in order to allow only sentences that have a low likelihood of containing anaphoric expressions (cf. Figure 3).

3.1 System description

The modified system is mainly build on the system described in (Schilder and Kondadadi, 2008), with two important modifications:

- The update summarization part of the main task demanded *supplemental features* that could capture whether the information is old or new. We added more features to our model in order to make this distinction.
- We introduced a *more aggressive filter* that allows only sentences that are similar to typical first sentences in a news article. Reviewing automatically generated summaries from previous years, we noticed that summaries that were scored highly often contain the first sentences. Note also that a baseline of selecting first sentences is difficult to beat for the multi-summarization task. In addition, we wanted to improve the overall linguistic quality of our summaries. Because first sentences normally do not contain anaphoric references or require references to preceding sentences, they satisfactorily meet this goal.

3.1.1 Features

The update system has the following features from FastSum: Topic title frequency, Topic description frequency, Content word frequency and Document frequency. In addition to these, we also used the following features.

Old Content Word Frequency: the relative content word frequency $p_c(t_i)$ of all *old* content words $t_{1..|s|}$ occurring in a sentence s. The *old* content word probability is defined as $p_c(t_i) = \frac{n}{N}$, where n is the number of times the word occurred in the *old* cluster and N is the total number of words in the *old* cluster: $\frac{\sum_{i=1}^{|s|} p_c(t_i)}{|s|}$

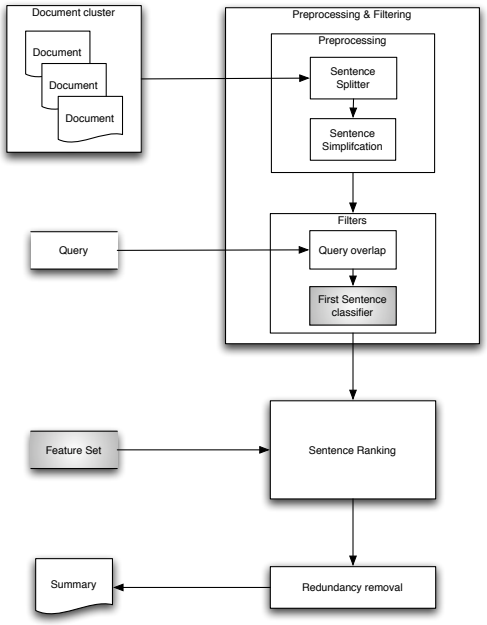

Figure 3: FastSum architecture for update summarization

Old Document Frequency: the relative document frequency $p_d(t_i)$ of all *old* content words $t_{1..|s|}$ occurring in a sentence s. The *old* document probability is defined as $p_d(t_i) = \frac{d}{D}$, where d is the number of *documents* the word t_i occurred in for the *old* cluster and D is the total number of documents in the *old* cluster: $\frac{\sum_{i=1}^{|s|} p_d(t_i)}{|s|}$

Old Entities: number of named entities in the sentence that already occurred in the old cluster.[3]

New Entities: number of new named entities in the sentence not already mentioned in the old cluster.

Old/New Entity Ratio: the ratio of number of unseen named entities in the sentence to the number of named entities in the sentence that were already seen.

[3]We extracted all mentions of persons, companies and locations using an in-house named entity tagger. We used exact string matching while matching the named entities.

New Words: number of new content words in the sentence not already mentioned in the old cluster.

Old Words: number of content words that already occurred in the old cluster.

Old New word ratio: the ratio of the number of unseen content words in the sentence to the number of content words in the sentence that appeared in the old cluster.

3.1.2 First sentence classifier

Problems that plague automatic summarization systems include dangling anaphoric expressions (e.g., pronouns, definite expressions) and incoherence of the extracted text due to missing rhetorical links. Discourse markers that point to nowhere (e.g., *therefore*) also reduce the readability of extractive summaries. In order to improve the linguistic quality of the extractive summaries that FastSum provides, we introduced a so-called "first sentence" classifier. This is motivated as follows. Instead of solving these problems by developing robust anaphoric resolution approaches or a rhetorical parser, we focussed on extracting sentences that are less likely to introduce these problems in the first place. First sentences in news articles often comprise the main facts of the article, are more focused and rarely contain anaphoric expressions. Hence, we developed a first sentence classifier in order to select sentences that look like sentences from the beginning of news articles in order to improve the linguistic quality of the extractive summaries. Note that we do not extract only first sentences from documents, for this year's baseline, as this might lose information from the rest of the document.

To extract the training data for this classifier, we used the AQUAINT-2 collection. We randomly selected 50,000 documents from this collection. For each document, the first sentence was added to the postive training data set and the rest of the sentences were added to the negative data set. We built a classifier between the positive and negative data sets using the following features:

Capitalized words: Number of capitalized words in the sentence normalized by the number of words.

Pronouns: Number of pronouns in the sentence normalized by the number of words.

Definite articles: Number of definite articles in the sentence normalized by the number of words.

Connector words: Is the first word a connector word? We used a list of connectors such as "Because", "Amid", "Therefore", etc.

Quotes: Does the sentence have quotes in it?

Words: Number of words in the sentence.

Testing our first sentence classifier on an unseen test set of another 50,000 sentences, we obtained an accuracy of 74.47%. Precision for first sentences was 70.73% and recall was 83.49%.

3.2 Results

The results from FastSum's participation in the update task are reported on below. We submitted three runs of which two were evaluated manually as well as automatically:

TOC1 (26) The full FastSum system with aggressive filtering using all features

TOC2 (52) The FastSum system after feature engineering via LARS plus aggressive filtering

TOC3 (69) A simple first sentence baseline with redundancy removal based on cosine similarity

We discuss how well our systems did in terms of linguistic quality, responsiveness, Pyramid score (Nenkova and Passonneau, 2004) as well as BE (Hovy et al., 2006) and ROUGE (Lin and Hovy, 2003) scores. We also share observations about the correlations between manual and automatic evaluation scores in the following sections.

Manual evaluation Summaries were manually evaluated using the following metrics: (a) *Linguistic Quality*: NIST assessors assigned an overall linguistic quality score to each of the automatic and human summaries. The summaries were scored on a scale from 1 (very poor) to 5 (very good) taking into account factors like non-redundancy, focus, structure and coherence, (b) *Responsiveness*: NIST assessors assigned an overall responsiveness score to each of the automatic and human summaries. The overall

responsiveness score is an integer between 1 (very poor) and 5 (very good) and is based on both the linguistic quality of the summary and the amount of information in the summary that helps to satisfy the information need expressed in the topic narrative, and (c) *Pyramid score*: NIST assessors created "Pyramid" from the four model summaries for each document set, and annotated peer summaries using the pyramid guidelines provided by Columbia University.

We received very high scores for linguistic quality and overall responsiveness but average scores for the pyramid evaluation. Our best run ranked 4^{th} among the peer systems for linguistic quality which indicates that the usage of the first sentence filter has a positive effect on the linguistic quality. For responsiveness, we ranked 8^{th} out of 58 peer systems. For the first sub-task, which is defined as query-based multi-document summarization, we even rank 1^{st} together with another system.

For the pyramid evaluation, we ranked 23^{rd}. It is unclear to us why there is such a difference between the Responsiveness and Pyramid score even though the two scores do correlate to each other, as shown in Figure 4.[4]

Automatic evaluation At TAC 2008, two automatic metrics were used: (a) *ROUGE*: NIST computed ROUGE-2 and ROUGE-SU4 scores by running ROUGE-1.5.5 with stemming but no removal of stopwords, and *BE*: NIST ran the BE-1.1 tools in order to obtain Basic Elements (BE) scores for systems and model summaries.

Our ROUGE scores were not as high relative to other systems as they were for the manual evaluation. We only ranked 28^{th} and 29^{th} for our two FastSum runs (52/26) out of 71 systems. Analyzing the correlation between the ROUGE-2 and Responsiveness score, two observations can be made. The Pearson coefficient is still high, but not as high as for Responsiveness vs. Pyramid score (0.8941). In addition, Figure 4 indicates that the top 22 systems that also had a Responsiveness score (ROUGE-2 scores > 0.08) show no correlation between ROUGE-2 and Responsiveness. The Pearson coefficient for these systems is 0.0687. Consequently, low ROUGE-2 scores can be seen as indication for low summarization performance, but high ROUGE-2 scores cannot differentiate good and very good performing systems.

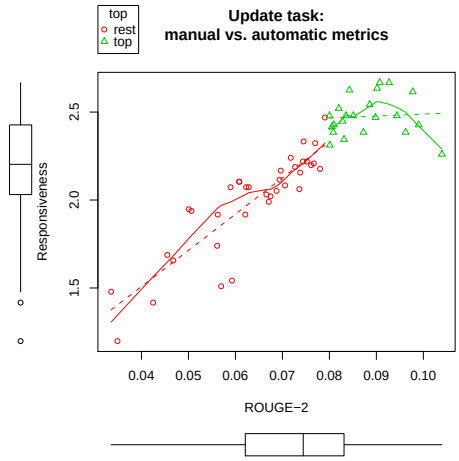

Figure 4: Responsiveness/ROUGE-2

For the BE evaluation, a similar picture emerged. TOC2 with optimized features ranked 27^{th}, TOC1 with the full feature set ranked 30^{th} and TOC3, our first sentence baseline, ranked higher than our other systems with 12^{th} place.[5]

Baseline We created our baseline by extracting first sentences from each document in the cluster and sorting them according to the document's timestamp. To eliminate any redundancy, a sentence was added to the summary only if the cosine similarity between the sentence and the summary was less than a threshold. We used 0.7 as the threshold in our experiments.

Interestingly enough, our baseline system which was not manually evaluated ranked 13^{th}, although the ROUGE-2 score still lies within the 95% confidence interval of our two other runs.

4 Opinion summarization

4.1 Related work

Initial notions on opinion-based summarization and its use in question answering was proposed by par-

[4]Pearson coefficient: 0.950684.

[5]Considering only the top 22 systems, one observes that the Pearson coefficient for BE and Responsiveness showed a weak correlation between the automatic and the manual evaluation metric: 0.4409.

ticipants in the ARDA MPQA initiative,[6] in the early 2000s (Cardie et al., 2003). It is significant to point out, however, that this work did not yet describe a method or system for implementing query-based sentiment summarization. By contrast, some of the first collective work on summarizing Question Answering for blogs occurred in the context of NTCIR in 2007 (Evans et al., 2007; Seki et al., 2007). In an English context, some of the first work performed in news and blog QA was conducted by (Somasundaran et al., 2007) and summarization in the same field by (Ku et al., 2006).

To our knowledge, this is the first operational method to summarizing the sentiment in general blog entries in a QA environment. It is also worth mentioning that this system is the first of its kind that is built to scale and exploits a robust set of features.

Related work has been performed, however, in specific domains within the blogosphere (Conrad and Schilder, 2007) as well as in specific professional topics within the application space (Conrad et al., 2008). In addition, seminal proposals have been circulated which offer more principled aproaches to leveraging search (Hearst et al., 2008) and data mining (Agarwal and Liu, 2008) in similar blog-based contexts.

4.2 Opinion FastSum system description

The overall workflow of the FastSum blog opinion summarization system is illustrated in Figure 6.

At a high level of abstraction, the principal components of the FastSum blog opinion summarizer include, in sequential order (see shaded boxes in Figure 5, from top to bottom, left to right):

1. HTML parsing and clean-up module;
2. Question sentiment and target analyzer;
3. FastSum target answer integration module;
4. Feature set for sentence ranking;

Whereas the question analyzer performs several types of analysis, it is the FastSum answer integrator that is responsible for both the quantity (length) and quality (filtered, scored, ranked, combined sentences) of the human-readable summary.

[6]ARDA – Advanced Research Development Agency, MPQA – Multi-Perspective Question Answering.

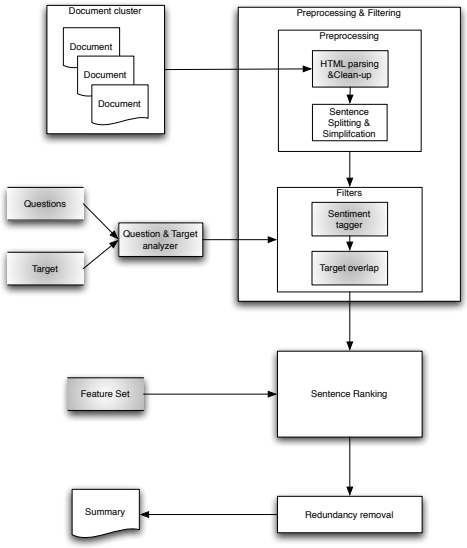

Figure 5: FastSum architecture for blog opinion summarization

4.3 Preprocessing & filtering

HTML parsing and Clean-Up We modified FastSum in order to process blogs by (a) utilizing an HTML parser to extract only text from the blog entries and (b) ignoring boilerplate language in the blogs (e.g., *Response by*). We used the Jericho htmlParser [7] for parsing the HTML documents. Deleting boilerplate language was achieved by a simple filter that (a) computed the density of capitalized words in a sentence[8] and (b) by matching a regular expression that contains frequently used language in blog entries.

Filtering Sentences were filtered according to their sentiment and whether the sentence was related to the target of the questions.

Sentiment tagging We implemented a sentiment polarity tagger largely based on unigram term lookup. While ultimately we believe that polarity tagging cannot be reduced to a context-insensitive word lookup task, we wanted to experiment with the impact of gazetteers, since such simplistic methods represent the largest part of the published literature, and they provide a baseline that more complex methods should benchmarked

[7]http://jerichohtml.sourceforge.net/doc/index.html

[8]Sentences that contain more than 50% capitalized words were automatically excluded.

Polarity	Precision	Recall	F-score (F1)
Positive	46.97%	51.67%	49.21%
Negative	59.52%	*33.78%*	43.10%
Neutral	61.99%	73.10%	67.09%
Overall	58.10%	58.10%	**58.10%**

Table 1: Component evaluation of sentiment polarity tagger

against.[9] We created gazetteers of positive and negative polarity-indicating terms based on the *General Inquirer* (Stone et al., 1966), extended it, and also eliminated some erroneous entries. We then proceeded to incorporate morphological variations of the words already in the gazetteer to improve coverage, and eliminated errors created in this process manually.[10] We noticed that many gazetteer entries were ambiguous in their status, namely whether sentiment polarity-bearing and *not* polarity-bearing. We thus decided to build two sets of polarity gazetteers, one as described above, and another one based on appreciable manual pruning, where all potentially ambivalent entries were eliminated to improve precision. For example, the entry *incompetent* was not pruned because it always expresses a negative sentiment, whereas *dependent* was removed from the second gazetteer since it may or may not be used in a neutral sense, depending on context.

The tagging itself was based on looking up tokens, counting positive and negative instances, and assigning a label as follows:

$$\begin{array}{lll} \texttt{NEGATIVE} & \text{if} & polarity < -1 \\ \texttt{NEUTRAL} & \text{if} & -1 \leq polarity \leq 1 \\ \texttt{POSITIVE} & \text{if} & polarity > +1 \end{array}$$

where $polarity = (\#PositiveTok - \#NegativeTok)/\#AllTok$.

We submitted two runs, one based on the gazetteer as described above (TOC1), and another one based on more aggressive manual gazetteer pruning performed on the negative gazetter (TOC2); see Table 2 for synopsis.

We developed a test set of 528 sentences randomly extracted from the BLOG06 subset that was provided by NIST as development data for the pilot task. Each sentence (segmented automatically by running the pre-processing pipeline of our system) was hand-labeled by a human reviewer with one of the three labels `NEGATIVE`, `NEUTRAL`, or `POSITIVE`. We performed a component-based evaluation specifically on the sentiment polarity tagger used in run TOC1. Since it is based on gazetteers, it does not require a training corpus. The results, given in Table 1, make it clear that recall issues for the negative class are a weak link of this approach. By contrast, the overall F1 score for the method was 58.1%.

Target matching Only those sentences whose polarity matches that of the question are considered. For those sentences that satisfy this condition, they are additionally examined for concordance with the target. Note that the target need not explicitly be present in the sentence under consideration as long as it is present in a decaying window centered on a target description. We matched words with the target via the Jaro Winkler similarity function in order to account for misspellings of names (e.g., *Guiliani*). We used the Cosine window function for assigning "targetness" scores to words following an identified target. Since we did not use any parsing, the window function allowed to connect targets and sentiment words even across sentence boundaries.

4.4 Question & Target analyzer

The most critical component of FastSum's blog opinion summarization application is located in the query sentiment analysis module. Several forms of analysis take place here. These include:

1. target/topic of the question identified;
2. polarity of the question determined;
3. type of question classified, if possible, which effectively is a stronger form of the evidence determined above.

This essential information is subsequently used as a type of filter against candidate answers present in the input data sentences delivered by the QA systems.

[9] We also implemented a simple negation detection; however, this was not used as it did not outperform a system without negation detection.

[10] This was not technically necessary, since non-words will almost never be looked up.

Sentiment Pilot Run	Pos. Entries	Neg. Entries	Type	Snippet Use	Pyramid F-Score
TOC1 (full gazetteer)	6,569	8,180	automatic	No	0.176
TOC2 (pruned gazetteer)	2,959	4,920	automatic	No	0.150

Table 2: Gazetteer statistics and sentiment summarization results

Sentence-based evidence that responds to the current question is accumulated across blog entries. FastSum is then deployed to score, rank and prune the evidence as necessary according to procedures described above; however, we did not use the first sentence classifier for the sentiment summaries.

It should also be noted that when multiple questions are posed for the same set of documents, where only the polarity is changed, a single summary is expected, and FastSum delivers but a single summary. This type of summary is generated based on TAC requirements, despite the fact that such a combination works against the strengths—the actual granularity and specificity—designed into FastSum.

4.5 Results

Our results in Table 2 show that TOC1, the run with the unpruned gazetteer, achieves a slightly better result overall. Individual performance varies considerably across queries, and two topics achieved a zero result. Overall our best sentiment summarization run ranked fourth in terms of Pyramid F-score out of the fully automatic systems among the 36 total evaluated runs. However, the associated ROUGE-2 score was very low.

Figure 6 shows how the different system types compare to each other in terms of F-score and ROUGE-2. There is an interesting difference between systems that used snippets and systems that did not use snippets. Overall, the scores for systems that did not use snippets are relatively low. Moreover, the Pearson coefficient for this subset of systems is again very low: 0.3018. The Pearson coefficient for systems that use snippets is 0.886 (excluding systems with manual interventions).

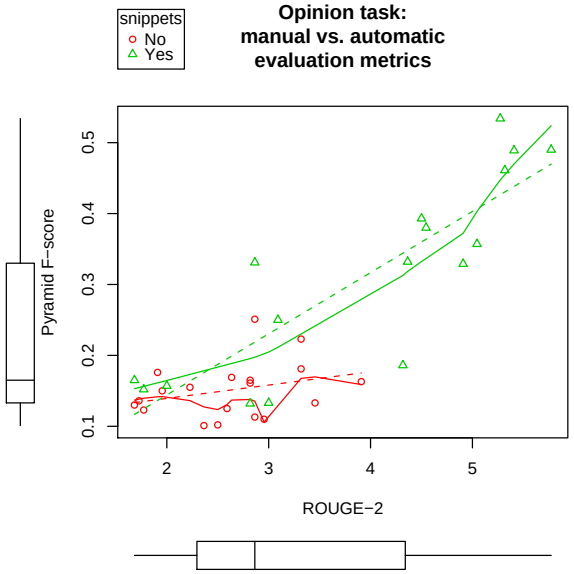

Figure 6: Pyramid F-score vs. ROUGE-2.

5 Conclusion and discussion

This year we modified the FastSum system for the main and pilot tasks contributing the following:

- Linguistic quality was improved by introducing a first sentence classifier. Our best system ranked 4^{th} for overall linguistic quality.
- Update summaries were generated via a regression SVM using features such as number of old/new entities.
- The optimal number of features for the regression SVM used by our first run was determined via a feature selection algorithm called LARS showing similar performance as the same system using the full set of features. The run with all features ranked 7^{th} for responsiveness, whereas the run with the optimized feature set ranked 8^{th}.
- We proposed a simple baseline for the update task that received high ROUGE scores.
- A sentiment consistency checker was added to the FastSum system for the pilot task.[11]

[11] Details on the sentiment analyzer will be covered by a poster.

Automatic evaluations Our findings regarding ROUGE and BE for the update task point to shortcoming of these automatic metrics. An automatic metric that takes into account linguistic quality, for example, may improve this correlation again.

Sentiment Pilot Task: a critique Like summarizing factual text, summarizing sentiments can be a time saving task, and sentiment continues to play an important role in a brand and attention oriented society. The Sentiment Pilot Task therefore represents a realistic use case, and the use of the BLOG06 corpus for this is highly appropriate. The manner in which the evaluation was set up, however, could be improved: the topics combined questions asking for positive or negative evidence with respect to the same topic (see Section 2.1), which implies that a system based on sentiment polarity consistency could not show its full strength. We therefore propose that in future tasks of this kind, positive and negative questions be evaluated separately.

Future work Even though we added features relevant to the update task, scores for this particular task were not as high as the scores for the query-based multi-document summarization. We thus want to investigate other features that capture the update part of this task more effectively.

For the sentiment summarization, we plan to compare the performance of unigram-gazetteers against methods that make use of context, since this initially requires the existence of n-gram authority files with sentiment phrases, we will explore how to use semi-supervised learning to bootstrap these resources.

Acknowledgments

We would like to thank our colleagues Dan Dyke for annotating our sentiment development data, Steve Rank and Kajsa Anderson for IT support, Marc Light for sharing data sampling scripts, and Khalid Al-Kofahi and Peter Jackson for supporting our TAC participation. We are also grateful to James Allan for related insightful discussions.

References

Nitin Agarwal and Huan Liu. 2008. Blogosphere: Research issues, tools, and applications. *KDD Explorations*, 10(1):19–29, June.

D. Bollegala, Y. Matsuo, and M. Ishizuka. 2007. Measuring Semantic Similarity between Words Using Web Search Engines. In *Proc. of 16th International World Wide Web Conference (WWW 2007)*, pages 757–766, Banff, Canada.

Claire Cardie, Janyce Wiebe, Theresa Wilson, and Diane Litman. 2003. Combining low-level and summary representations of opinions for multi-perspective question answering. In *Proceedings of the AAAI Spring Symposium on New Directions in Question Answering*, pages 20–27.

Jack G. Conrad and Frank Schilder. 2007. Opinion mining in legal blogs. In *Proceedings of the 11th International Conference on Artificial Intelligence and Law (ICAIL07)*, pages 231–236, Palo Alto, CA. ACM Press.

Jack G. Conrad, Jochen Leidner, and Frank Schilder. 2008. Professional credibility: Authority on the Web. In *Proceedings of the Second Workshop on Credibility on the Web (WICOW 2008)*, Napa Valley, CA. ACM Press.

B. Efron, T. Hastie, I.M. Johnstone, and R. Tibshirani. 2004. Least angle regression. *Annals of Statistics*, 32(2):407–499.

David Kirk Evans, Lun-Wei Ku, Yohei Seki, Hsin-Hsi Chen, and Noriko Kando. 2007. Opinion analysis across languages: an overview of and observations from the NTCIR6 opinion analysis pilot task. In *Proceedings of the Workshop on Cross-Language Information Processing*, volume 4578 (Applications of Fuzzy Sets Theory) of *Lecture Notes in Computer Science*, pages 456–463.

Marti A. Hearst, Matthew Hurst, and Susan T. Dumais. 2008. What should blog search look like? In *Proceedings of the Workshop on Search and Social Media (SSM08)*, Napa Valley, CA. ACM Press.

E. H. Hovy, C.-Y. Lin, L. Zhou, and J. Fukumoto. 2006. Automated summarization evaluation with Basic Elements. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*. ELRA.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142. ACM SIGKDD, ACM.

Lun-Wei Ku, Yu-Ting Liang, and Hsin-Hsi Chen. 2006. Opinion extraction, summarization and tracking n news and blog corpora. In *Proceedings of the Spring Symposium on Computational Approaches to Analyzing Weblogs (AAAI-CAAW 2006)*, Palo Alto, CA. AAAI Press.

S. Li, Y. Ouyang, W. Wang, and B. Sun. 2007. Multi-document summarization using support vector regression. In *Proceedings of DUC 2007, Rochester, USA*.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL)*, pages 71–78, Morristown, NJ, USA. Association for Computational Linguistics.

A. Nenkova and R. Passonneau. 2004. Evaluating content selection in summarization: The Pyramid method. In *Proceedings of the HLT-NAACL Conference*. ACL.

Frank Schilder and Ravikumar Kondadadi. 2008. FastSum: Fast and Accurate Query-based Multi-document Summarization. In *Proceedings of ACL-08: HLT, Short Papers*, pages 205–208, Columbus, Ohio, June. Association for Computational Linguistics.

Yohei Seki, David Kirk Evans, Lun-Wei Ku, Hsin-Hsi Chen, Noriko Kando, and Chin-Yew Lin. 2007. Overview of opinion analysis pilot task at NTCIR-6. In *Proceedings of the Workshop Meeting of the National Institute of Informatics (NII) Test Collection for Information Retrieval Systems (NTCIR)*, pages 265–278.

Swapna Somasundaran, Theresa Wilson, Janyce Wiebe, and Veselin Stoyanov. 2007. QA with attitude: Exploiting opinion type analysis for improving question answering in on-line discussions and the news. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM 2007)*.

Philip J. Stone, Dexter C. Dunphy, Marshall S. Smith, and Daniel M. Ogilvie. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. The MIT Press, Cambridge, MA, USA.