

# The University of British Columbia at TAC 2008

**Gabriel Murray** and **Shafiq Joty** and **Giuseppe Carenini** and **Raymond Ng**  
Department of Computer Science  
University of British Columbia  
Vancouver, BC V6T 1Z4 Canada

## Abstract

In this paper we describe the University of British Columbia's participation in the Text Analysis Conference 2008. This work represents our first submission to the DUC/TAC series of conferences, and we participated in both the summarization tasks: the main update task as well as the pilot task on summarizing blog opinions. We describe our systems in detail and describe our performance in the context of all submitted systems.

## 1 Introduction

In the first year of the Text Analysis Conference, there were two main summarization tasks, both of which the University of British Columbia participated in: a main update task carried over from the conference's previous incarnation as the Document Understanding Conference (DUC) as well as a novel pilot task on summarizing opinions from blogs. This latter task was particularly interesting to our research group, as we have been focusing on the summarization of informal conversation media such as meetings, emails and chats, including detecting subjectivity in these domains. We accordingly weight this paper towards a description of the pilot task system and its performance.

## 2 Update Task

The update task involves summarizing newswire data. For each topic, the system is presented with two clusters of documents. The system must first summarize cluster A, then summarize cluster B with

the knowledge that cluster A has already been seen. It is therefore paramount to reduce the amount of redundancy in the output summaries when summarizing the B clusters. In the sections below we describe our approach to this problem.

### 2.1 Features Used

One aim of our update summarization system is to discriminate between different types of redundancy. Penalties for redundancy have long been a part of multi-document summarization methods such as Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998). For example, a sentence's overall MMR score is simply calculated as its relevance score minus its redundancy score, where the redundancy score is derived by comparing the candidate sentence with the already-extracted sentences. Penalizing redundancy too much, however, can lead the summary output to stray from the intended topic.

While penalizing redundancy can lead to lexical diversity in the final summary, it is not the case that favouring redundancy necessarily leads to a summary containing sentences with nearly identical content. The reason is that an initially extracted sentence might be selected because of terms that closely overlap with the topic, but it may contain other terms that build new links between remaining sentences and the topic. That is, favouring redundancy can be a form of implicit query expansion. To give a toy example, if a topic asks "Which games do dogs like to play?", the most relevant sentence may be A below, but the inclusion of A in the summary-so-far also makes sentence B suddenly relevant:

- A: My dog loves to play in the park with this

ragged old ball.

- B: My terrier loves chasing balls and frisbees in the park.

The terms “park” and “ball” were, in effect, added to the topic query, thereby making sentence B relevant. This query expansion would be lost in an MMR-style framework. Naturally, it is undesirable for a system to extract nearly-identical sentences, and so a threshold can be included for the maximum tolerated redundancy.

For the update task we submitted two systems, described below. The first system is an unsupervised approach, and the second uses many of the same features in a supervised, machine-learning framework. Before describing the individual systems, we present the shared feature set here.

There are several query-relevance metrics used. The score *cos1* is simply the cosine between the query vector and the candidate sentence vector. We carry out query expansion by using the Infomap latent semantic analysis (LSA) tool<sup>1</sup> and finding the most closely related words to the query terms. The score *cos2* is then the cosine between the expanded query vector and the candidate sentence vector.

The score *olap* is derived by summing the scores for terms that are shared by both the candidate and the original query vector, then dividing by the sum of query vector values. The metric *nonolap* is derived by summing the candidate sentence term scores, subtracting the total of the term-weights shared by the sentence and the query, and dividing this subtotal by the sum of the candidate sentence term scores. This score measures what percentage of the sentence’s term-scores are non-overlapping with the query, i.e. how much the sentence would expand the topic if added to the summary-so-far.

The metric *red1* measures the candidate sentence’s redundancy with the already-extracted sentences, using the maximum cosine of the candidate and the set of extracted sentences. The score *red2* measures the sentence’s similarity to the summaries generated for the previous cluster. For the A cluster, these scores are always 0.

The *docscore* metric measures the similarity between the document as a whole and the query. This

feature	weight
cos1	1
cos2	1
olap	1
dscore	1
poscore	1
nonolap	0.6
red1	0.6
red2	-1

Table 1: Feature Weights

is motivated by the intuition that a sentence should be more likely to be extracted if the document in which it occurs is very relevant. The score *posit* indicates the position of the candidate sentence in its source document, with 1 representing the beginning of the document and 0 representing the end. This is motivated by the fact that the first paragraphs of an article tend to provide a high-level overview of the subject.

In cases where a candidate sentence is 30 words or longer in length, we break the sentence into its constituent clauses using the Sundance parser (Riloff and Phillips, 2004), and treat each clause as a separate candidate sentence. This is to ensure that our final output summary does not contain only three or four sentences.

## 2.2 Unsupervised Update System

For the unsupervised system, an overall sentence score is derived by summing over all of the features, with each feature having an associated weight as listed in Table 1. The only penalty is therefore the *red2* score, measuring overlap with the summary sentences from the previous cluster. The sentences are then ranked according to this combined score and extracted until the maximum summary length is reached. The system ID for the unsupervised update system is 29.

## 2.3 Supervised Update System

The supervised system uses all of the previously described features and more. The *olap* and *nonolap* scores are calculated using both the original query and expanded query. We also calculate a variety of redundancy features for the previous cluster, comparing the candidate sentence to both the previous cluster summaries and the previous cluster document sentences as a whole. We also introduce a

<sup>1</sup><http://infomap-nlp.sourceforge.net>

feature *sharecon* that compares each candidate sentence to all of the other candidate sentences with higher *cosI* scores, and measures the sum of term-weights for terms that are shared between these sentence pairs but are not contained in the query. So if sentence A has a higher *cosI* score than sentence B, we sum the term-weights for all terms that are shared between the two sentences but are not contained in the query, and assign this as the *sharecon* score for sentence B. This helps measure query relevance for sentences that might not share exact terms with the query but are nonetheless related to the topic.

There were two central challenges with our supervised summarization system. The first is that we did not possess sentence-level summarization annotation for training purposes for this task. The second was the challenge in utilizing dynamic features in a static system. We address these two concerns in turn.

For training our update task summarizer, we needed gold-standard extractive/non-extractive annotations. While we had data for the 2007 DUC Update Task, this did not contain sentence-level annotations. We decided to derive these class labels automatically by measuring n-gram overlap between each sentence from the 2007 task documents with the relevant human abstracts, using ROUGE-2 and ROUGE-SU4. We then considered the top-scoring sentences according to ROUGE to be the summary sentences for that topic (using a summary length of 200 words), with the remaining sentences being members of the non-extractive class. While this would lead to noisy class labels, we predicted that ROUGE would give a rough approximation of sentence informativeness for training purposes.

The second challenge was in deciding how to capture dynamic features in a static summarization framework. In the unsupervised system, the *redI* scores change every time that a sentence is included in the growing summary. In other words, *redI* is a dynamic feature. This is straight-forward in an unsupervised algorithm but more complex in a machine-learning framework. For the supervised system, we calculated the *redI* scores for each sentence by first ranking all candidate sentences according to *cosI* scores, then calculating for each sentence its *redI* score for all of the sentences ranked higher. So for a given candidate sentence, we simply consider

its summary-so-far to be the set of sentences with higher *cosI* scores.

The system ID for the supervised update system is 55.

## 2.4 Results

Both of our update task systems performed approximately average in comparison with the 71 submitted systems. For the A cluster documents, the average Pyramid score for all participant systems was 0.26. Our unsupervised system averaged 0.245 and the supervised system averaged 0.232. For overall responsiveness, the average for all systems was 2.32. Our systems averaged 2.1 and 2.2, respectively. For overall responsiveness, our systems were ranked 46th and 41st out of all participant systems.

Both of our systems fared substantially better on summarizing the B clusters, in terms of system ranking. The average Pyramid score was 0.21 overall, and our unsupervised and supervised systems averaged 0.23 and 0.22, respectively. The average responsiveness for all participant systems was 2.03, and our systems averaged 2.06 and 1.98. They were ranked 28th and 35th overall. For ROUGE-SU4, our systems hovered around the overall average of 0.11, with averages of 0.11 and 0.10.

Our expectation was that our supervised system would perform slightly better, particularly on ROUGE since ROUGE was used for determining the training labels using the 2007 data, but this was not the case. We suspect that the class labels were simply too noisy for our supervised system.

## 3 Blog Opinion Task

For the blog opinion summarization task, we submitted two systems. Both systems share the same subjectivity components, which we describe in the next section. The systems differ primarily in their use of query expansion and named entity detection and matching. For both systems, we chose not to use the available QA snippets.

### 3.1 Subjectivity Features

Our blog summarization systems use two components for subjectivity detection. These are described in turn below.

### 3.1.1 Extraction Patterns for Positive and Negative Subjectivity

The first subjectivity component is based on identifying positive and negative extraction patterns in the manner of Riloff et al. (Riloff and Wiebe, 2003). This component utilizes the Sundance parser and the AutoSlog-TS algorithm for identifying relevant extraction patterns from data that is annotated for subjectivity. This algorithm takes as input a corpus, a portion of which is annotated for the domain of interest, e.g. positive subjectivity. All of the extraction patterns from the corpus are extracted, where an extraction pattern is essentially a partial lexical instantiation of a syntactic pattern. For example, in the work of Riloff et al. in detecting sentences about terrorism, significant extraction patterns included “sympathizers of <np>” and “<subj> kills bystanders.” The extraction patterns that occur more often in the data subset of interest are considered to be highly indicative of that domain. For example, we can find the extraction patterns that occur often in sentences with positive subjectivity but rarely in the corpus as a whole.

Because we do not possess blog data that is annotated for positive and negative subjectivity, we had to learn these positive and negative extraction patterns using a different corpus from the meetings domain. A portion of the AMI corpus<sup>2</sup> has been annotated with a large amount of subjectivity information (Wilson, 2008), and we hypothesized that the informal, conversational and often disfluent nature of meeting speech may share interesting characteristics with blog data.

We applied the AutoSlog-TS algorithm to the 20 annotated AMI meetings, automatically learning the extraction patterns for the positive and negative dialogue acts (DAs). The extraction patterns are scored in several different ways, and we chose to consider an extraction pattern to be a significant indicator of that domain if it occurred more than 2 times and had a posterior probability of at least 0.65. These are rather low thresholds, but using stricter criteria resulted in very few extraction patterns being retrieved.

One interesting result is that there are far more extraction patterns returned for positive DAs than for

Positive Patterns	Negative Patterns
<subj> BE easy	<subj> BE boring
<subj> BE better	<subj> BE bad
<subj> liked	<subj> avoid
change <dobj>	worry about <np>
<subj> agree	<subj> BE confusing

Table 2: Examples of Positive and Negative Patterns

negative DAs. We hypothesize that this is because participants in meetings will tend to couch negative sentiments in indirect language or euphemisms, whereas positive sentiments may be expressed more directly and use more typical repeated patterns. 185 positive extraction patterns were found, but only 41 negative patterns.

A few of the extraction patterns are not only domain-specific, but specific even to the AMI scenario task, but most are general. For example the top two positive patterns are “<subj> BE good”, with a probability of 0.833 and 36 total occurrences, and “<subj> BE easy,” with a probability of 0.731 and 26 total occurrences. The top two negative extraction patterns are “problem with <np>,” with a probability of 1 and six total occurrences, and “<subj> BE problem,” with a probability of 0.80 and 10 total occurrences. Table 2 lists some of the top positive and negative extraction patterns.

Having determined positive and negative extraction patterns, we can then score candidate sentences according to how many patterns match the sentence and how well the polarity of the sentence matches the polarity of the query.

### 3.1.2 Semantic Orientation Calculator

The second subjectivity tool used is the semantic orientation calculator (SO-CAL) (Taboada et al., 2008), originally developed for analyzing online product reviews and movie reviews. It works by combining a lexicon of subjective keywords with contextual cues that shift the polarity of the keywords. The keywords have associated values ranging from -5 to 5, i.e. from very negative to very positive. SO-CAL outputs a single score in the same range, indicating the subjectivity type of the document.

<sup>2</sup><http://www.amiproject.org>

Because SO-CAL was developed for online reviews, its lexicon is quite small and specialized. We supplemented the existing keywords with keywords gleaned from the MPQA corpus<sup>3</sup>. The MPQA keywords did not have the associated values that SO-CAL requires, so we mapped “weakly positive” and “weakly negative” keywords to 1 and -1, respectively, while “strongly positive” and “strong negative” keywords were given scores of 5 and -5, respectively.

SO-CAL typically generates overall document scores, but we used the calculator to generate scores for individual sentences, so that every candidate sentence is given a score between -5 and 5.

### 3.2 Blog Opinion System 1

The first blog summarization system is the more advanced of the two, featuring several preprocessing and query expansion components. These are described in turn below. The system ID for our first blog system is 3.

#### 3.2.1 Question Decomposition and Expansion

Question Decomposition involves breaking a compound question into its subparts. We look for the conjunctions to break the questions into two or more subquestions. We find the first-sense synonyms (using WordNet) of the nouns and verbs in a question. We expand the questions by adding these synonyms. For example the question: “What motivated positive opinions of CARMAX from car buyers” becomes “What motivate or actuate or propel or move or prompt or incite positive opinion or sentiment or thought or view or persuasion of CARMAX from car or auto or automobile or machine or motorcar buyers or purchaser or empton or vendee” after expansion.

#### 3.2.2 Named Entity Tagging

Named entities are terms that refer to certain entity. For example “Canada” refers to a country, “Vancouver” refers to a city. We used the OAK named entity tagger<sup>4</sup> to tag the questions with named entities. OAK uses a set of 150 different named entity tags. We capture the named entities of the question to compare it later with the named entities of the document sentence.

<sup>3</sup><http://www.cs.pitt.edu/mpqa/databaserelease>

<sup>4</sup><http://nlp.cs.nyu.edu/oak/>

#### 3.2.3 Named Entity Overlap Measure

For each document sentence we extract the named entities in the same way as we did for the questions. We measure the named entity overlap by counting the number of named entities common in them.

#### 3.2.4 Cosine Similarity Overlap Measure

The sentences are represented as sentence vectors based on the tf\*idf counts of the words. The questions are also represented as question vectors in the same way. We measure the the cosine similarity overlap of a sentence with a question by computing the angle between the sentence vector and the question vector as follows:

$$\theta = \cos^{-1} \frac{\vec{q} \cdot \vec{s}}{\|\vec{q}\| \times \|\vec{s}\|}$$

We measure the cosine similarity of a sentence with the original question as well as with the decomposed and expanded questions of the original question. Hence we have two cosine similarity measures for each document sentence: 1. *CosSimOrg*: that computes the similarity with the original question and 2. *CosSimExp*: that computes the similarity with the decomposed and expanded questions of the original question.

#### 3.2.5 Sentence Ranking

For each question of a target we rank the sentences. We have four different measures for each of the sentences:

1. *Subj*: The subjectivity score of the sentence, averaging the extraction pattern and SO-CAL scores.
2. *NE*: The named entity overlap score of the sentence.
3. *CosSimOrg*: The Cosine similarity overlap score with the original question.
4. *CosSimExp*: The Cosine similarity overlap score with the decomposed and expanded questions of the original question.

We want the sentences to be important with respect to its subjectivity and at the same time relevant to the question. We set a threshold  $T$  for this. Our ranking formula is:

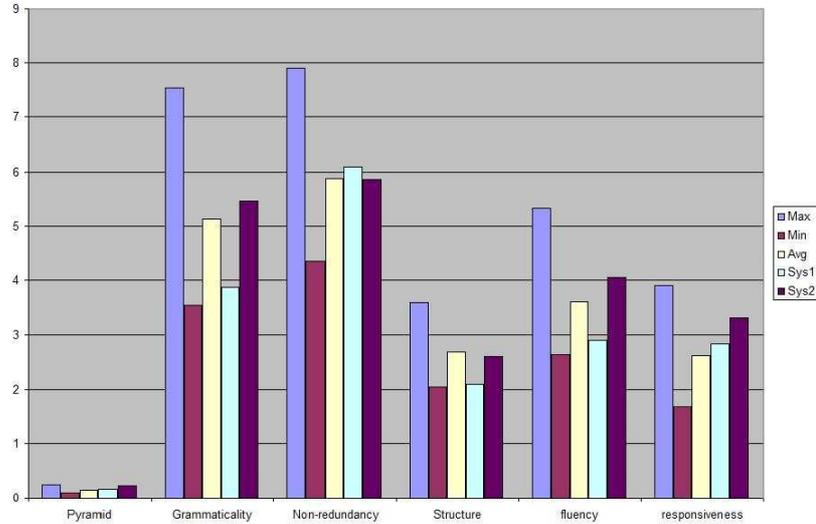


Figure 1: Blog Task Results for Non-Snippet Systems

$if(CosSimOrg > T) :$

$$score = Subj + NE + CosSimOrg + \frac{3}{4} \times CosSimExp$$

$else :$

$$score = NE + CosSimOrg + \frac{3}{4} \times CosSimExp$$

In this way, we prevent the sentences that have low relevancy with the question from being extracted simply because they have a high subjectivity score.

### 3.2.6 Redundancy Checking and Producing Answer

The answer length is maximum 7000 characters for each question but we do not include all the top ranked sentences upto 7000 characters in the final answer. The sentence which has score greater than a fixed threshold ( $T=1.5$ ) will be included in the final answer if it passes through a redundancy checker. The redundancy checker checks the cosine similarity of a candidate summary sentence (to be added) with each of the sentences which are already in the summary. The sentences for which the similarity measure is below a certain threshold (0.75) will be included in the final answer. In this way we produce answers for each of the questions in the target.

### 3.3 Blog Opinion System 2

Our second blog system is essentially a simplified version of the first. This system does not use query

expansion or named entity recognition and matching. There are essentially three components to a sentence score. The first score is derived from the subjectivity tools described previously, representing the polarity of the sentence. There are then two relevance metrics: cosine with the query and cosine with the topic. If the relevance metrics fall below a certain threshold, the overall sentence score is derived using only the relevance metrics and no subjectivity score. This is to prevent the case where a sentence is selected because of a very high subjectivity score but nevertheless has low relevance scores.

The system ID for the second system is 25.

### 3.4 Results

Of all submitted systems, 19 in total did not use the QA snippets. Among those 19 systems, our two submitted systems fared very well according to the various evaluation metrics. For Pyramid scores, the average for all 19 systems was 0.15. Our first system, UBC1, averaged 0.16 and our second system, UBC2, averaged 0.22. This latter score was the second highest of the 19 systems. For overall responsiveness, both of our systems were again above the average of 2.61. UBC1 averaged 2.82 while UBC2 averaged 3.32, which tied for third place among the 19 systems.

The most surprising result was that UBC2, the simpler of our two submitted systems, performed

much better than UBC1. We hypothesize that this is because of the query expansion component that is present in UBC1, and that query expansion is actually detrimental for this task. The topic questions are very specific, asking for opinions about entities such as “Jiffy Lube” and “Carmax,” and it may be the case that query expansion causes sentences to be extracted that are only tangentially related to the topic. Interestingly, UBC2 is also much better than UBC1 on criteria of grammaticality and fluency.

The only criteria for which UBC2 is below the average for all 19 systems are non-redundancy, in which case UBC2 is just below average and UBC1 is actually better, and the structure/coherence score.

Figure 1 gives the scores for the maximum, minimum and average performance of the 19 non-snippet systems, in comparison with our UBC1 and UBC2 systems.

## 4 Conclusion

For TAC 2008, UBC submitted a total of four systems - two for each summarization task. For the update task, our systems performed average, with the unsupervised system slightly better than the supervised system.

For the blog opinion task, our systems performed well, with the UBC2 system being among the best of all submitted non-snippet systems according to Pyramid and responsiveness metrics. We plan to explore novel methods for detecting subjectivity in blogs and other informal conversations such as meetings and emails.

## 5 Acknowledgments

Thank you to Maite Taboada and Julian Brooke of Simon Fraser University for the use of their SO-CAL code.

## References

- J. Carbonell and J. Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proc. of ACM SIGIR Conference on Research and Development in Information Retrieval 1998, Melbourne, Australia*, pages 335–336.
- E. Riloff and W. Phillips. 2004. An introduction to the sundance and autoslog systems.

- E. Riloff and J. Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Proc. of EMNLP 2003, Sapporo, Japan*.
- M. Taboada, K. Voll, and J. Brooke. 2008. Extracting sentiment as a function of discourse structure and topicality. *JNLE*, to appear.
- T. Wilson. 2008. Annotating subjective content in meetings. In *Proc. of LREC 2008, Marrakech, Morocco*.