Update Summary Update

Terry COPECK, Anna KAZANTSEVA, Alistair KENNEDY,
Alex KUNADZE, Diana INKPEN, Stan SZPAKOWICZ
School of Information Technology and Engineering
University of Ottawa
800 King Edward Avenue
Ottawa, Ontario, Canada K1N 6N5
{terry,ankazant,akennedy,diana,szpak}@site.uottawa.ca
akunadze@opentext.com

Abstract

We submitted runs from two different systems for the update summary task at TAC 2008. One system used *Roget's Thesaurus* to determine semantic relatedness for the purpose of summary construction. The other system employs a variety of heuristics, including an innovative use of the topic headline in the assessment of semantic similarity among sentences. Our submission to the opinion task used only the provided text snippets. Work continues here on a deeper semantic representation. We will also update the SCU-marked corpus with data from the 2008 conference.

1 Introduction

The participation of the University of Ottawa's NLP research group in the NIST-sponsored summarization challenges helps us organize and structure our activities in the area of text summarization, and lets us evaluate our systems' performance in the broader scope of the summarization community. Even more than in previous years (Copeck *et al*. 2006, 2007), NIST's evaluation effort in the latest cycle has suited to our research agenda very well.

NIST limited resources for manual evaluation had previously permitted a DUC participant to submit only one set of summaries per task. This need not have affected groups who worked together to develop a single system. To a team like ours – composed primarily of graduate students each of whom pursued a unique line of research – it meant merging the results of separate and heterogeneous systems into a single submission to go in under our name. This is what we have repeatedly done, averaging the sentence ratings computed by each constituent system to arrive at an overall sentence ranking that in some manner represented all researchers' contributions equally. This was not optimal; the evaluation of performance was necessarily imprecise.

TAC 2008 opened the door to multiple task submissions. At the University of Ottawa, two graduate students undertook this year to rate sentences in the test corpus on their suitability for an update summary. Sections 2 and 3 present their systems. For the first time, separate results were submitted without modification. The evaluation that TAC assessors performed is thus highly pertinent to each of the systems. We hope that NIST will continue to accept multiple runs from participants in a given task.

Our team also submitted a run for the 2008 pilot task. Section 4 presents the design of opinion summary selection, not influenced by any ongoing research agenda. We also regularly update the corpus marked with Summary Content Unit [SCU] (Copeck *et al*. 2007) with each year's new data. The corpus is available to TAC participants on request to NIST. Note that as DUC/TAC annual tasks evolve, so too does the content of the SCU-marked corpus. That is because the basis on which documents are annotated with SCUs has changed over the years. Topics that reflect *simple query-focused summaries* can now be considered a closed class, with that task replaced by the current requirement to produce update summaries.

2 Summarizing with *Roget's Thesaurus* and SCU-marked corpora

This system employs *Roget's Thesaurus* as a tool for text summarization. The motivation for using *Roget's* for this purpose comes from Kennedy and Szpakowicz (2008) where the 1911 and 1987 versions of *Roget's* were shown to perform equally well on tasks such as measuring semantic relatedness between words and synonym identification. *Roget's* was also shown to be a good tool for enhancing vector based representation of sentences and measuring sentence similarity – this technique will be described later on. Initially the goal of this line of research was to generate a system that could either replace or be used to enhance the graph-matching system of Nastase and Szpakowicz (2006), which we used in previous years.

We rely on our SCU-marked corpus, in which sentences known to be relevant to a particular query are labelled with the appropriate SCU identifiers. This information makes the corpus an excellent tool for evaluating summarization systems that perform sentence extraction; it can also be used for developing systems that identify redundancy. We explore here both these uses.

Our final system – see section 2.6 – uses the 1911 *Roget's Thesaurus* to enhance a *tf.idf*-based ranking of sentence relevance. A multi-layered perceptron network is trained to identify redundancy in text. These methods are enhanced with a few small heuristics.

2.1 The SCU-marked corpus

We used the SCU-marked corpus for testing the systems. The corpus has been generated from the data of previous DUC competitions. Each sentence in a summary submitted to DUC is labelled with the SCUs it contains. A sentence can contain 0, 1 or more than 1 SCU. These SCUs have weights from 1 to 8 for the 2005 data and 1 to 4 for the 2006 data. Since a sentence can have multiple SCUs, it is possible for its total SCU score to be very high. These sentences are then mapped back into the original document set.

Sentences found in a summary that contained 0 SCUs are negative examples. Sentences found in a summary that contained 1 or more SCUs are positive examples. Sentences that never appeared in a summary are unlabeled (Note: unlabeled is not the same as neutral.) The 2005 data contain 1187 positive and 1490 negative examples. The 2006 data contain 988 positive and 1368 negative examples. There are many unlabeled examples, but for the most part they are ignored during evaluation.

2.2 Roget's Thesaurus

Roget's is a hierarchical thesaurus. There are altogether nine levels in the hierarchy, from top to bottom:

- Class
- Section
- Subsection
- Head Group
- Head
- Part of Speech
- Paragraph
- Semicolon Group
- Words

The words are always at the leaves of this structure. They include nouns, verbs, adjectives, adverbs and several other less common parts of speech such as interjections. Clearly, this structure differs significantly from *WordNet*'s.

The Open *Roget's* Project (rogets.site.uottawa.ca) has recently released a free version of *Roget's Thesaurus*. We use both this system and an analogous system based on the proprietary 1987 data; the latter is not in the public domain.

2.3 Sentence Ranking

Our sentences ranking is based on their predicted relevance to the query: how likely it is that a sentence contains a SCU from the SCU-marked corpus. We test several methods of predicting the relevance of a sentence. Most of them rank sentences by their similarity to the query. The query may contain several questions and instructions (expected contents of answers), but our methods attempt to match the query as a whole, not individual questions and instructions.

2.3.1 Graph Matching

Graph matching was the backbone of our last year's summarizer (Nastase and Szpakowicz 2006). Graph matching works by extracting two kinds of features from the queries and sentences. The first kind of feature is relationships. The corpus is parsed using MiniPar (Lin 1998) and dependency pairs are found. Two words are related if both appear in the same dependency pair.

The second feature is made up of all noun and verb unigrams from the query and sentences. We expanded these unigrams by selecting synonyms of the nouns and verbs from their dominant sense in *WordNet* 2.0.

Each sentence gets a score based on word and relationship overlap. The number of overlapping words is S_W. The number of overlapping relationships is S_R. The score for the sentence is $S_W + weight*S_R$. The *weight* we use is 15.

For testing purposes we re-implemented that system (its developer has since left our group). This might not be its perfect replication. In fact, in Nastase and Szpakowicz (2006) another methods that mixes graph matching and path matching was slightly better.

2.3.2 TF.IDF

Queries are weighted with term frequency only. In this system we treat each sentence as its own document. Inverse document frequency is the logarithm of the number of sentences (S) divided by the number of sentences containing term S_t.

Term frequency *tf* is simply a count of how many times a term appears in that sentence. Each term is weighted with *tf*idf*. We remove 980 stop words, as well as

$$idf = \log\left(\frac{|S|}{|S_t|}\right)$$

punctuation, from both the queries and the sentences. Cosine similarity determines the distance between the query and each sentence. This is similar to what was done in Radev *et al*. (2004).

2.3.3 Enhanced TF.IDF

This section describes a framework for enhancing *tf.idf* using lexical resources – *WordNet* and *Roget's Thesaurus*. A similar sentence representation has been tested in Kennedy and Szpakowicz (2008). The query and sentences are represented by terms as well as concepts from *WordNet* or *Roget's*. Each word *w* is given a score of 1. Each sense of *w* found in the thesaurus (*Roget's* or *WordNet*) is given a score of 1/X, where X is the number of *w*'s senses. 1/X is added to each of that word sense's ancestors in the resource. In *WordNet*, this means that each hypernym of the word sense has its score increased by 1/X. In *Roget's* this means that 1/X is added to the semicolon group, paragraph, ..., class. This creates a vector of terms as well as concepts (from *Roget's* or *WordNet*) that are weighted with term frequency. Inverse document frequency is calculated for all words, as well as concepts, and the vectors are weighted with *tf.idf*. This tf.idf enhancement has been tested three times, using concepts that come only from Roget's 1987, only from Roget's 1911 and only from WordNet.

2.3.4 Baselines

For comparison purposes, we experimented with two baseline methods. One is to not bother with ranking the sentences on any criteria. This is essentially ranking in collection order; sentences are selected in the order they appear in the document set.

The second baseline is to rank based on sentence length. This should be a higher baseline since longer sentences are more likely to contain SCUs.

2.3.5 Evaluation

We use average precision to evaluate the systems. Average precision is calculated by first sorting all the sentences. Next, iterate through the list from highest to lowest: calculate the precision at each positive instance and average those precisions.

$$AveP = \frac{\sum_{r=1}^{N}\left(\Pr ecision(r) * rel(r)\right)}{number\,of\,relevant\,sentences}$$

Precision(r) is the precision up to sentence *r* and *rel(r)* is a binary function, 1 if sentence *r* is relevant (has a SCU), and 0 otherwise (has no SCU). We calculate average precision for every set of queries and documents, and then take the average over each of them for a given year. This is a macro average of the average precision. We report results for the 2005/2006 data in Table 1 and for the 2007 Update Summary task in

Table 2. Table 2 contains document sets A, B and C as well as their average.

System	Precision: 2005	Precision: 2006
Graph matching	.504	.472
tf.idf	.517	.518
Roget's 1911	**.576**	.525
Roget's 1987	.565	.523
WordNet	.560	**.526**
Random	*.431*	*.463*
Sentence Length	*.569*	*.525*

Table 1: Macro average precision of each system for the 2005 and 2006 data

System	A	B	C	Avg
Graph matching	.534	.448	.471	0.484
tf.idf	**.652**	.550	.544	0.582
Roget's 1911	.644	.560	**.582**	**0.595**
Roget's 1987	.639	**.567**	.555	0.587
WordNet	.639	.531	.583	0.584
Random	*.588*	*.460*	*.451*	0.500
Sentence Length	***.675***	*.490*	*.581*	0.582

Table 2: Macro average average precision for the 2007 Update Data

From this data it appears that *tf.idf* enhanced with the 1911 *Roget's Thesaurus* gives the best results on the 2005 and 2007 data sets, and is a close second behind *WordNet* on the 2006 data set.

Selecting sentences based on sentence length performs extremely well on every data set giving it a very high score. Intuitively the more words you have in a sentence, the higher the chance that it contains a SCU. Sentence length performs deceivingly well. That is because often the longest sentences can have 60 or more words. Thus, in a 100-word summary it may be impossible to create a summary with more than 1 or 2 sentences. Graph matching also tends to favour longer sentences. As a result, these two methods will create summaries with a few long sentences, while other methods will create summaries with many shorter sentences.

2.4 Sentence Novelty

Sentence novelty is to do with the ability to detect whether two sentences contain the same information. This will be used for two purposes. The first is to identify and eliminate sentences that describe topics that appeared in previous summaries (update summaries). The second is to eliminate redundancy between sentences in the same summary. These experiments use the SCU-marked corpus and run algorithms from Weka (Witten & Eibe 2005). All sentences are represented using *tf.idf*-weighted terms and the 1911 *Roget's Thesaurus* concepts.

2.4.1 Data Set

We assume that two sentences with the same SCU identifier contain some overlapping information. We also assume that all other SCU-labelled sentences in the same document set without that SCU identifier have no overlapping information. (This may not be always true, since two sentences with different SCU identifiers could have some overlapping information if that information were not assigned a SCU identifier.) The data set is made up of all pairs of SCU-labelled sentences from the 2005, 2006 and 2007 update summary data sets. The positive to negative example ratio is approximately 1:10.

2.4.2 Features

A vector of terms and a vector of concepts from *Roget's* and *WordNet* represent each sentence. Let the sentence vectors be denoted *S1* and *S2*.

A total of seven features are extracted from these vectors. The first is just the cosine distance between the two sentence vectors. The other six features have to do with measuring content overlap.

If an element *v* appears in a vector, it has a non-zero weight. *v* may appear in both vectors, with weights *weight(v,S1)* and *weight(v,S2)*. The total weight of a sentence is

$$sentenceWeight = \sum_{v \in S1} weight(v, S1)$$

Feature 2 is the proportion of *S1* that overlaps with *S2* and feature 3 is the proportion of *S2* that overlaps with *S1*. The following formula is for vector *S1*.

$$Overlap = \frac{\sum_{\substack{v \in S1 \\ v \in S2}} \min(weight(v,S1), weight(v,S2))}{\sum_{v \in S1} weight(v,S1)}$$

Features 4 and 5 are the proportion of the weights made up by the difference in weight for all nodes that appear in both vectors.

$$Differences = \frac{\sum_{\substack{v \in S1 \\ v \in S2 \\ weight(v,S1) > weight(v,S2)}} (weight(v,S1) - weight(v,S2))}{\sum_{v \in S1} weight(v,S1)}$$

Features 6 and 7 are the proportion of the total weight that comes from nodes that appear only in one vector, but not the other.

$$Exclusive = \frac{\sum_{\substack{v \in S1 \\ v \notin S2}} weight(v,S1)}{\sum_{v \in S1} weight(v,S1)}$$

We ran tests with all seven features on a variety of ML algorithms from Weka. The 2005 and 2006 data were used for training and the 2007 data for testing.

Algorithm	class	prec	recall	F score	ROC Area
Naive Bayes	Pos	.448	.361	.400	.72
	Neg	*.872*	*.907*	*.889*	
Bayes Net	Pos	.381	.472	**.442**	.733
	Neg	*.884*	*.84*	*.861*	
LibSVM	Pos	.913	.022	.043	.511
	Neg	*.831*	*1*	*.907*	
Logistic	Pos	.688	.145	.239	.737
	Neg	*.847*	*.986*	*.911*	
Multilayer Perceptron	Pos	.735	.101	.177	**.738**
	Neg	*.841*	*.992*	*.911*	
J48	Pos	.657	.092	.161	.681
	Neg	*.839*	*.99*	*.909*	

Table 3: Precision, recall and area under the ROC curve for redundancy detection

The highest F-score for the positive class was for Bayes Nets with the 2005 and 2006 data as training data. The highest ROC Area was for Multilayered Perceptron. For the negative class the F-score was almost always near 0.9. We decided to use Multilayered Perceptron as our algorithm, since in addition to having the highest ROC value it also had a much higher precision for the positive class than other methods. By doing this we are reducing the risk of misclassifying a sentence as redundant at the cost of occasionally having some redundant information. Thus, we lower the risk of throwing out a good sentence, yet still have some redundancy checking.

2.5 Heuristics

Three heuristics were applied to this system before submission. The first is to eliminate sentences with 5 words or fewer. Although we produce no numbers to back this up, we have observed that very short sentences rarely contain any useful information and often are in fact grammatically inadequate sentence fragments.

The second heuristic is to eliminate sentences that contain quotations. Once again we do not perform any tests to back the underlying claim up, but we have observed that such sentences rarely fit well into a summary. This is because the identification of the speaker of the quote may not be included in the sentence and so is lost in the summary.

The third heuristic is to only consider the top 50 ranked sentences in a document set. This is done both to save time and improve accuracy.

2.6 Final System

Our final system employs a greedy algorithm to select the most relevant sentences. The algorithm selects the next sentence that fits into the summary. If a sentence is too long, or if it is judged to be redundant by our multilayered perceptron, then it is skipped over. This can happen with either a sentence that already exists in the summary, or one that exists in the previous document set (for the update summary).

The actual ranking of the sentences is based on their score of similarity to the query, normalized by the length of the sentence. This tends to favour many short sentences over a few long sentences. From several user

tests we found that normalizing based on length favours responsiveness over readability.

When generating the updated summary, comparing each sentence in the second document set against every sentence in the first document set can take a prohibitively long time. That is why we select the top 50 ranked sentences from that document set to represent it. All sentences are compared against just those 50 sentences. This works under the assumption that any sentences in any document set that are ranked below 50 are irrelevant to the summary and so there should ideally be no need to test them for redundancy.

The order of the sentences is determined using a lexical chain algorithm. We choose the ordering that maximizes the sum of the scores for each lexical chain found. This is implemented using the algorithm in Jarmasz and Szpakowicz (2003). It is not completely clear that doing this will drastically improve the readability of the summary, but it should not produce results any worse than random ordering. It should also be noted that every single ordering of sentences cannot be tested in reasonable time, so we only produce our best guess at the optimal ordering.

After the summary has been produced, an anaphora resolution module replaces pronouns with their referents.

2.7 Results

The results from evaluation put our system in the bottom half of the pack on most measurements. One area where this system does really well is the average number of repetitions. A score of 0.635 was found for this system, below the average of 0.791. On all other measures this system was only slightly behind the average.

3 Summarizing Using Headlines and Multiple Heuristics

The update task at TAC 2008 calls for summarizing a collection of documents with the assumption that the reader is already familiar with the background information in a separate collection on the same topic. The task is more challenging than those of the previous years. Effectively, it requires that the facts included in the summary not only be salient, but that they also not overlap with the user's previous knowledge of the topic.

The data for the update task consist of 48 topics, with two collections of documents available for each topic. The first collection contains ten newswire articles with which the reader is already familiar (further *background collection*). The second collection of documents contains another ten documents that update the previous one (further *updating collection*). Each article is accompanied by a headline. The summary of the background collection has only to reflect the important facts found in those documents. The summary of the updating collection needs to be such that it does not repeat what the user would know after reading the complete background collection.

Our system creates purely extractive summaries without any editing. In order to select salient and novel sentences we rely on several shallow heuristics. We use the headlines that accompany each article and approximate the salience of each sentence by computing lexical overlap with the corresponding headline. Another indicator of salience is the *tf.idf* metric. To give preference to sentences that express new facts, we modify the *tf.idf* measure so as to reward the terms that have not appeared in the background collection. Two scores that reflect these properties are combined, and we select sentences with the highest ranks to create a 100-word summary. In addition, we penalize sentences that are too similar to the sentences already found in either the background or the updating summaries.

3.1 System Description

The data available for training and/or parameter tuning consists of ten topics used in the DUC 2007 update task. We use this small corpus to select the best heuristics and to adjust the available parameters. The best-performing combination is then applied to the test data.

Preprocessing. Before proceeding to sentence selection, all documents are pre-processed in the following manner. First, the documents are tokenized using the BALIE tool (Nadeau) and stop-words are removed. Next, the texts are stemmed using the Lovins

stemmer (Lovins 1968), which is made available as a part of Weka through its API (Witten and Frank 2005).

Creating background summaries. In order to select sentences that are good candidates for inclusion into the summary, we rank all sentences using a combination of two scores: *tf.idf* score and lexical similarity with the corresponding headline.

The *tf.idf* score of a *term* rewards terms frequent in a collection at hand but rare in the whole corpus. tf_{ij} is the frequency of a term w_i in the topic d_j, df_i is the number of topics in the collection where w_i occurs at least once and N is the number of token types in the compete corpus. In our setting, *tf.idf* score of a term gives an idea of how central the term is in the collection at hand.

$$\text{tf.idf} = \begin{cases} (1+\log(\text{tf}_{ij}))\log\frac{N}{\text{df}_i} & \text{if } \text{tf}_{ij} \geq 1 \\ 0 \text{ if } \text{tf}_{ij} = 0 \end{cases}$$

The *tf.idf* score of a *sentence* is the sum of scores of all its terms normalized by the sentence length.

The second score that approximates the importance of a sentence is lexical overlap with the headline of the corresponding article. Effectively, the headline already is a summary of the article, so it is only logical that it describes the most important facts in it.

$$\cos(\vec{s},\vec{h}) = \frac{\vec{s} \bullet \vec{h}}{|\vec{s}||\vec{h}|} = \frac{\sum_{i=1}^{n} s_i h_i}{\sqrt{\sum_{i=1}^{n} s_i^2}\sqrt{\sum_{i=1}^{n} h_i^2}}$$

The similarity with the headline is measured using the cosine metric (Manning and Schütze 1999, p. 300): where vectors $\vec{s}$ and $\vec{h}$ correspond to the candidate sentence and the headline one.

Removal of redundancies. In addition, we use cosine similarity to avoid repetition in the summaries. For each candidate sentence, we measure how similar it is to those already found in the summary. If the similarity value exceeds 0.5, the sentence is skipped.

Creating updating summaries. The overall algorithm for creating updating summaries is quite similar to that for creating background ones.

The first exception is the calculation of *tf.idf* score. In order to reward terms not found in the background collection, we multiply their *tf.idf* score by a factor of two. This process rewards the new terms in proportion to their original score (terms with low *tf.idf* remain at the bottom of the list).

When checking for redundancies, we look for sentences that are too similar to those found in either the updating or the background summary of the collection at hand. We use the same metric and the same threshold to achieve this end.

3.2 Results

The summaries are evaluated using several metrics. There is no obvious way of combining them into a single ranking. Looking across the available metrics, however, gives one a good idea of the system's performance. Ours is ranked 37th and 35th when measuring overall responsiveness for background and updating summaries respectively (corresponding to values of 2.23 and 1.92). The modified pyramid scores are 3.37 for background and 2.46 for updating summaries, corresponding to rank 42 in both cases. Our system's ROUGE-2 and ROUGE-SU4 scores are rather poor.

These results are not very good and appear to suggest that using naïve heuristics by themselves is not sufficient to create summaries of good quality.

4 Summarizing Opinion

In addition to summarizing successive subsets of documents on a given topic for update information, TAC 2008 asked participants to produce summaries of opinion. Raw materials for this task included not only the usual collection of documents on a topic and one or more questions about it to direct the summarization effort, but a new intermediate resource. It was a list of text fragments – *snippets* produced by QA systems or human annotators – which address a specific topic information need, together with the identifier of the document from which each snippet was extracted.

We considered the effort invested in producing this pre-selection of potentially relevant text fragments, and the fact that this is a pilot task to which we had not had the opportunity to devote much attention. We decided to base our system on processing the snippet list and ignore the base document collections from which its entries were derived. The pilot task thus became for us

one of attempting to produce a summary of the snippets on a topic which best answered the question about it – a markedly easier task.

4.1 The Process

The first step in accomplishing this was to clean up the snippets, which were taken from fairly "dirty" blog documents. Inspection of the first test topic identified a number of divergences from proper English syntax[1]—punctuation spaced away from the token to which it was attached, repeated characters, incorrectly rendered punctuation, and so on. We corrected 17 syntactic irregularities on one pass, and three lexical ones on a second. Capitalization was an issue in the first topic snippet set, and tokens appearing all in upper or lower case were converted to mixed case when instances were found elsewhere in the blog to guide this operation. A by-product of the cleaning operation was to eliminate duplicates from the hash set of snippets when these appeared in the original data or as a result of correcting variants to the canonical well-formed form.

The second stage of processing selected the snippets to compose the summary. We used four heuristics to eliminate candidates from contention. 1) Snippets over 500 characters in length were removed (too long to be focused on the topic), as were 2) those which were approximately identical (where match was determined approximately using the Perl `amatch` library), 3) those which are subsumed by another snippet, and finally, 4) those which appeared to incorporate more than one sentence.

This pruning operation reduced the set of candidate snippets to a total length of about 4,000 characters on average across all 22 topics, well under the 14,000 characters allowed (7,000 characters were accepted per squishy question and each topic had two squishy questions). Lacking any better basis on which to order these snippets, the submitted summary was written out ordered shortest to longest, on the speculation that shorter snippets would contain a higher proportion of pertinent information.

4.2 The Outcome

NIST evaluated the opinion pilot in a manner comparable to that applied to the main task. Six measures were employed. Five were assessed manually: content, responsiveness and fluency/readability (this subsumed grammaticality, non-redundancy and structure/ coherence). A submission's pyramid F-score was computed automatically. Each participant's average score for most[2] of these measures are reported in the file `OpSumm08.avg_scores`. According to its scores for the three non-repeated measures (F-score, overall responsiveness and overall fluency/readability), our submission was ranked first in a tie with that of participant 9. Averaging all scores in the file including the three submeasures of fluency/readability ranked us 8th. On the single score of responsiveness which we ourselves deem key, we scored 4th.

This rather surprising performance likely highlights the importance of the snippet selection procedure in producing opinion summaries. It certainly is not due to the simple correction operations and filtering heuristics of which our system is composed – though cleaning up the snippets probably did improve their readability.

5 Future Work

Goals for the next year are to continue to find occasions when we can conduct experiments in which our team itself judges summary responsiveness and fluency. That should allow a system's developer to improve its sentence selection process either through trial and error or by iterative refinement.

We will continue to update the corpus of SCU-marked topics with new material as it becomes available, and to use it to guide future development of summarization systems at the University of Ottawa as appropriate.

[1] We classified our system as "manual" on the submission form because we based its design on inspection of the first topic in the test data set. Processing by the system, once implemented, was wholly automatic.

[2] Content *per se* is absent, while measures are included both of overall fluency/readability, and three of its constituents: grammaticality, non-redundancy and structure-and-coherence.

Finally, we hope to move towards employing selection algorithms which are based on deeper semantic knowledge of a text as a team member's research in this area comes to fruition.

Acknowledgment

Partial support for this work comes from the Natural Sciences and Engineering Research Council of Canada.

References

Copeck, Terry, Diana Inkpen, Anna Kazantseva, Alistair Kennedy, Darren Kipp and Stan Szpakowicz. 2007. Catch What You Can. *Proceedings of the Workshop on Automatic Summarization* (DUC 2007), HLT/NAACL-2007.

Copeck, Terry, Diana Inkpen, Anna Kazantseva, Alistair Kennedy, Darren Kipp, Vivi Nastase and Stan Szpakowicz. 2006. Leveraging DUC. *Proceedings of the Workshop on Automatic Summarization* (DUC 2006), HLT/NAACL-2006.

Copeck, Terry and Stan Szpakowicz. 2005. Leveraging Pyramids. *Proceedings of the Workshop on Automatic Summarization* (DUC 2005), HLT/EMNLP-2005.

Jarmasz, Mario and Stan Szpakowicz. 2003. Not As Easy As It Seems: Automating the Construction of Lexical Chains Using Roget's Thesaurus. *Proceedings of the 16th Canadian Conference on Artificial Intelligence (AI 2003),* Halifax, Canada, June, 544–549.

Kennedy, Alistair and Stan Szpakowicz. 2008. Evaluating Roget's Thesauri. *Proceedings of* ACL-2008, 416-424.

Lin, D. 1998. Dependency-based Evaluation of MINIPAR. In *Proceedings of the workshop on the Evaluation of Parsing Systems*, First International Conference on Language Resources and Evaluation

Lovins, J.B. 1968. Development of a Stemming Algorithm. Mechanical Translation and Computational Linguistics, 11, 22-31.

Manning, C.D and Schütze, H. 1999. *Foundations of statistical natural language processing*, MIT Press, Cambridge, MA.

Nadeau, D. Multilingual Information Extraction from Text with Machine Learning and Natural Language Techniques. Technical report. sourceforge.net/docman/display_doc.php?docid=26784&group_id=124581.

Nastase, Vivi and Stan Szpakowicz. 2006. A Study of Two Graph Algorithms in Topic-driven Summarization. *Proceedings of the Workshop on Graph-based Algorithms for Natural Language* Processing (TextGraphs2006), HLT/NAACL-2006.

Radev, Dragomir R., Hongyan Jing, Małgorzata Styś, and Daniel Tam. Centroid-based summarization of multiple documents. Information Processing and Management, 40:919–938, December 2004.

Witten, Ian H. and Eibe Frank. 2005. *Data Mining: Practical machine learning tools and techniques, 2nd ed.* Morgan Kaufmann, San Francisco, 2005.