Evaluation of a Sentence Ranker for Text Summarization Based on Roget's Thesaurus

Alistair Kennedy[1] and Stan Szpakowicz[1,2]

[1] School of Information Technology and Engineering
University of Ottawa, Ottawa, Ontario, Canada
{akennedy,szpak}@site.uottawa.ca

[2] Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland

Abstract. Evaluation is one of the hardest tasks in automatic text summarization. It is perhaps even harder to determine how much a particular component of a summarization system contributes to the success of the whole system. We examine how to evaluate the sentence ranking component using a corpus which has been partially labelled with Summary Content Units. To demonstrate this technique, we apply it to the evaluation of a new sentence-ranking system which uses *Roget's Thesaurus*. This corpus provides a quick and nearly automatic method of evaluating the quality of sentence ranking.

1 Motivation and Related Work

One of the hardest tasks in Natural Language Processing is text summarization: given a document or a collection of related documents, generate a (much) shorter text which presents only the main points. A summary can be generic – no restrictions other than the required compression – or query-driven, when the summary must answer a few questions or focus on the topic of the query. Language generation is a hard problem, so summarization usually relies on *extracting* relevant sentences and arranging them into a summary. While it is, on the face of it, easy to produce *some* summary, a *good* summary is a challenge, so evaluation is essential. We discuss the use of a corpus labelled with Summary Content Units for evaluating the sentence ranking component of a query-driven extractive text summarization system. We do it in two ways: directly evaluate sentence ranking using Macro-Average Precision; and evaluate summaries generated using that ranking, thus indirectly evaluating the ranking system itself.

The annual Text Analysis Conference (TAC; formerly Document Understanding Conference, or DUC), organized by The National Institute of Standards and Technology (NIST), includes tasks in text summarization. In 2005–2007, the challenge was to generate 250-word query-driven summaries of news article collections of 20–50 articles. In 2008–2009 (after a 2007 pilot), the focus has shifted to creating *update summaries* where the document set is split into a few subsets, from which 100-word summaries are generated.

<line>As opposed to the international media hype that surrounded last week's flight, with hundreds of journalists on site to capture the historic moment, Airbus chose to conduct Wednesday's test more discreetly.<annotation scu-count="2" sum-count="1" sums="0"><scu uid="11" label="Airbus A380 flew its maiden test flight" weight="4"/><scu uid="12" label="taking its maiden flight April 27" weight="3"/></annotation></line>
<line>After its glitzy debut, the new Airbus super-jumbo jet A380 now must prove soon it can fly, and eventually turn a profit.<annotation scu-count="0" sum-count="3" sums="14,44,57"/></line>
<line>"The takeoff went perfectly," Alain Garcia, an Airbus engineering executive, told the LCI television station in Paris.</line>

Fig. 1. Positive, negative and neutral sentence examples for the query "Airbus A380 – Describe developments in the production and launch of the Airbus A380"

1.1 Summary Evaluation

One kind of manual evaluation at DUC/TAC is a full evaluation of the *readability* and *responsiveness* of the summaries. Responsiveness tells us how good a summary is; the score should be a mix of grammaticality and content.

Another method of manual evaluation is *pyramid evaluation* [1]. It begins with creating several reference summaries and determining what information in them is most relevant. Each relevant element is called a Summary Content Unit (SCU), carried in text by a fragment, from a few words to a sentence. All SCUs are marked in the reference summaries and make up a so-called pyramid, with few frequent SCUs at the top and many rarer ones at the bottom. In the pyramid evaluation proper, annotators identify SCUs in peer summaries. The *SCU count* tells us how much relevant information a peer summary contains, and what redundancy there is if a SCU appears more than once. The *modified pyramid score* measures the recall of SCUs in a peer summary [2].

1.2 The SCU-Labelled Corpus

Pyramid evaluation supplies fully annotated peer summaries. Those are usually extractive summaries, so one can map sentences in them back to the original corpus [3]. Many sentences in the corpus can be labeled with the list of SCUs they contain, as well as the score for each of these SCUs and their identifiers. [3] reported that 83% of the sentences from the peer summaries in 2005 and 96% of the sentences from the peer summaries in 2006 could be mapped back to the original corpus. A dataset has been generated for the DUC/TAC main task data in years 2005–2009, and the update task in 2007.

We consider three kinds of sentences, illustrated in Figure 1. First, a *positive* example: its <annotation> tag shows its use in summary with ID 0, and lists two SCUs: with ID 11, weight 4, and with ID 12, weight 2. The second sentence – a *negative* example – has a SCU count of 0, but is annotated because of its use in summaries 14, 44 and 57. The third *unlabelled* sentence was not used in any summary: no annotation. The data set contains 19,248 labelled sentences from a total of 91,658 in 277 document sets.[1] The labelled data are 39.7% positive.

Parts of the SCU-labelled corpus have been used in other research. In [4], the 2005 data are the means for evaluating two sentence-ranking graph-matching algorithms

[1] There is one document set from each of the 2005–2007 main tasks, three sets from the 2007 update task and two sets each from the 2008–2009 main tasks.

for summarization. The rankers match the parsed sentences in the query with parsed sentences in the document set. For evaluation, summaries were constructed from the highest-ranked sentences. The sum of sentence SCU scores was the score for the summary. One problem with this method is that both labelled and unlabelled data were used in this evaluation, thus making the summary SCU scores a lower bound on the expected scores of the summary. Also the method does not directly evaluate a sentence ranker on its own, but rather in tandem with a simple summarization system.

In [5], an SVM is trained on positive and negative sentences from the 2006 DUC data and tested on the 2005 data. The features include sentence position, overlap with the query and others based on text cohesion. In [6], the SCU-labelled corpus is used to find a baseline algorithm for update summarization called Sub-Optimal Position Policy (SPP), an extension of Optimal Position Policy (OPP) [7]. In [8], the corpus from 2005–2007 is used to determine that summaries generated automatically tend to be query-biased (answer a query) rather than query-focused (select sentences to maximize overlap with a query).

2 Sentence Ranking

We compare a new method of sentence ranking against a variety of baselines, which we also describe here. The proposed method uses a function for measuring semantic distance between two terms, available through Open *Roget's*, a Java implementation of the 1911 *Roget's Thesaurus* ⟨rogets.site.uottawa.ca⟩. We also had access to a version with proprietary 1987 *Roget's* data, which allowed us to compare newer and older vocabulary. *Roget's* is a hierarchical thesaurus comprising 9 levels; words are always at the lowest level in the hierarchy.[2]

2.1 *Roget's* SemDist

The *SemDist* function based on *Roget's* was originally implemented for the 1987 version [9] but has recently also been made available for the 1911 version [10]. We use a modified version of *SemDist* pairs of words are scored 0..18, farthest to closest, where 18 is given when a word is compared with itself. The distance returned by *SemDist* is simply the edge distance between two words in *Roget's Thesaurus*, subtracted from 18. *SemDist* is used to generate a score $score(S)$ indicating the similarity of a sentence S to the query Q. The distance between each word $w \in S$ is measured against each word $q \in Q$.

$$score(S) = \sum_{q \in Q} \max\left(SemDist(w, q) : w \in S\right)$$

$score(S)$ ranks sentences by relevance to the query. This system can be implemented without *SemDist*: let a score be either 0 or 18. We also ran an experiment with this method – called Simple Match (SM) – and the methods using the 1911 and 1987 Thesauri. Stop words (we used the stop list from [11]) and punctuation are removed

[2] The 1911 version has around 100,000 words and phrases (60,000 of them unique), the 1987 version – some 250,000 (100,000 unique) words and phrases.

from both the queries and the sentences. This method tends to favour long sentences: a longer sentence has more chances of one of its words having a high similarity score to a given word in the query q_i. We see the effect of this tendency in an indirect evaluation by summary generation (Section 3.2).

2.2 Term Frequency – Inverse Sentence Frequency (*tf.isf*)

Term Frequency - Inverse Document Frequency (*tf.idf*) is widely used in document classification. We rank sentences, not documents, so we talk of Term Frequency – Inverse Sentence Frequency (*tf.isf*). The query is also treated as a single sentence (even if it has a few actual sentences). Again, stop words and punctuation are ignored. Cosine similarity is used to determine the distance between the query and each sentence. This is similar to what was done in [12].

2.3 Other Baselines

We include three baseline methods for comparison's sake. One is simply to rank sentences based on the number of words in them; referred to as *Length*. The second method is to order the sentences randomly; we label this method *Random*. The last method, *Ordered*, is to not bother ranking the sentences on any criteria: sentences are selected in the order in which they appear in the data set.

3 Evaluation and Results

We now discuss the evaluation of the systems from Section 2: *SemDist* 1987 and 1911, Simple Match and *tf.isf*, plus the three baseline methods. They will undergo two kinds of evaluation with the SCU-labelled corpus. In Section 3.2 we discuss an evaluation similar to what is performed in [4], but we exclude unlabelled sentences from the evaluation and only generate summaries with up to 100 words. We already noted a drawback: rather than directly evaluate the sentence ranker, this evaluates a ranker in tandem with a simple sentence-selection system.

We also need a method of determining how well a sentence ranker works on its own. To do this, in Section 3.1 we evaluate our ranked list of sentences using Macro-Average Precision. This will give us an overall score of how well the sentence ranker separates positive from negative sentences. We choose Macro-Average instead of Micro-Average, because the score each sentence receives depends on the query it is answering, so scores are not comparable between document sets. Again unlabelled sentences are excluded from this evaluation.

3.1 Direct Evaluation with Macro-average Precision

The calculation of average precision begins by sorting all the sentences in the order of their score. Next, we iterate through the list from highest to lowest, calculating the precision at each positive instance and averaging those precisions.

$$AveP = \sum_{r=1}^{N} Prec(r) \times Rec(r)$$

$Prec(r)$ is the precision up to sentence r, $Rec(r)$ – the change in recall [13]. The macro-average of the average precision is taken over all document sets, thus giving the macro-average precision, reported in Table 1.

Table 1. SCU Rankings for data

	SD-1911	**SD-1987**	**SM**	***tf.isf***	**Length**	**Random**	**Ordered**
Score	**0.572**	0.570	0.557	0.521	0.540	0.445	0.460

The differences between the systems are not very high: *SemDist* 1911 scores only 5.1% higher than *tf.isf*. The improvement of *SemDist* 1911 over the *Random* and *Ordered* baselines is more noticeable, but the *Length* baseline performs very well. Nonetheless, it can clearly be seen from these results that the two *Roget's SemDist*-based methods perform better than the others. There are a total of 277 document sets in the whole data set, which is a suitably high number for determining whether the differences between systems are statistically significant. A paired t-test shows that the two *SemDist* methods were superior to all others at $p < 0.01$, but the difference between the two *SemDist* methods was not statistically significant. The SCU-labelled corpus provides us with a way to prove that one sentence ranker outperforms another.

One possible problem with this evaluation approach is that it does not take sentence length into account. Long sentences are more likely to contain a SCU simply by virtue of having more words. An obvious option for evaluation would be to normalize these scores by sentence length, but this would actually be a different ranking criterion. Were we to modify the ranking criteria in this way, we would find that *tf.isf* outperforms *SemDist* 1911, 1987 and Simple Match. That said, favouring longer sentences is not necessarily a bad idea when it comes to extractive text summarization. Each new sentence in a summary will tend to jump to a different topic within the summary, to the detriment of the narrative flow. Selecting longer sentences alleviates this by reducing the number of places where the flow of the summary is broken.

All these sentence-ranking methods could be also implemented with *Maximal Marginal Relevance* [14], but once again that is simply a different ranking criterion and can still be evaluated with this technique.

3.2 Indirect Evaluation by Generating Summaries

A second evaluation focuses on indirectly evaluating sentence rankers through summary generation. We demonstrate this on a fairly simple summary evaluation system. To build the summaries, we take the top 30 sentences from the ranked list of sentences. Iterating from the highest-ranked sentence to the lowest in our set of 30, we try to add each sentence to the summary. If the sentence makes the summary go over 100 words, it is skipped and we move to the next sentence.

Table 2 shows the results. We report the total/unique SCU score, total/unique SCU count and the number of positive and negative sentences in the 277 summaries generated. Unique SCU score is probably the most important measure, because it indicates the total amount of unique information in the summaries.

Table 2. Total SCU scores and counts in all summaries

System	**Total Score**	**Unique Score**	**Total SCUs**	**Unique SCUs**	**Positive Sentences**	**Negative Sentences**
SemDist 1911	2,627	**2,202**	1,067	907	530	**396**
SemDist 1987	2,497	2,126	1,023	874	515	398
Simple Match	**2,651**	2,200	**1,083**	**923**	542	466
tf.isf	2,501	2,122	1,001	864	**572**	721
Length	1,336	1,266	678	567	286	258
Random	1,726	1,594	762	676	438	854
Ordered	1,993	1,709	855	741	491	771

Simple Match and *SemDist* 1911 perform better than the other methods, one leading in the total SCU score and one leading in the unique SCU score. That said, the difference in terms of SCU scores and SCU counts between systems is very small. The most significant differences can be seen in the number of positive and negative sentences selected. Since *tf.isf* does not favour longer sentences, it is natural that it would select a larger number of sentences. Counted as the percentage of *tf.isf*'s selection of positive sentences, the 1987 *SemDist* method, the 1911 method and Simple Match give 90%, 93% and 95% respectively. The difference in the number of negative sentences is more pronounced. The 1987 and 1911 *SemDist* methods have just 55% as many negative sentences as *tf.isf* while Simple Match methods has 65%. This sort of evaluation shows that the ratio of positive to negative sentences is the highest for the *SemDist*-based methods and Simple Match. This supports the findings in Section 3.1. In fact, the percentage of positive sentences selected by *SemDist* 1911, 1987 and Simple Match was 57%, 56% and 54% respectively, while *tf.isf* had only 44%.

This indirect evaluation showcases the downside of relying too heavily on sentence length. The *Length* baseline performs very poorly on almost every measure; even the *Random* baseline beats it on all but negative sentence count. The percentage of sentences selected using *Length* is about 42% of how many were selected using any of the methods which do not favour longer sentences (*tf.isf*, *Random* and *Ordered*), averaging just 1.96 sentences per summary. By comparison, Simple Match had about 78% and the two *SemDist* methods contained about 71% of the number of sentences used by *tf.isf*, *Random* and *Ordered*.

Our method of evaluating summary generation can also estimate redundancy in a summary by examining the number of total and unique SCUs. Both *SemDist* methods, Simple Match and *tf.isf* had about 85% as many unique SCUs as total SCUs. This is comparable to the baseline methods of *Length*, *Random* and *Ordered* which had 84%, 89% and 87% as many unique SCUs as total SCUs respectively. There is little redundancy in the summaries we generated, but redundancy is tied to summary length. As a quick experiment, we ran the 1911 *Roget's* SemDist function to generate summaries of sizes 100, 250, 500 and 1,000. We found that the percentage of unique SCUs dropped from 85% to 73% to 62% to 52%. This shows the need for such a redundancy-checking system, and clearly the SCU-based corpus can be a valuable tool for evaluating it.

4 Conclusions and Discussion

We have shown two methods of evaluating sentence ranking systems using a corpus partially labelled with Summary Content Units. This evaluation is quick and inexpensive, because it follows entirely from the pyramid evaluation performed by TAC. As long as TAC performs pyramid evaluation, the SCU-labelled corpus should grow without much additional effort. We have also shown that, despite their individual drawbacks, our direct and indirect methods of evaluating sentence selection complement each other. Evaluating sentence-ranking systems using Macro-Average Precision allows us to determine how good a sentence-ranking system is by taking every labelled sentence in the document set into account. Because of the large number of document sets available, it can be used to determine statistical significance in the differences between sentence-ranking systems. The drawback could be the favouring of simplistic methods of selecting longer sentences. The indirect evaluation through summary generation cannot be fooled by systems selecting long sentences. It also provides us with a means of measuring redundancy in summaries. Its drawbacks are that it only evaluates a sentence ranker as it is used for generating a summary in one particular way.

The fact that we ignore unlabelled sentences rather hurts this as an overall evaluation of a summarization system. Were a summarization system to select sentences in part because of their neighbours or location in a document, we could not guarantee that that sentence would be labelled. If, however, sentences are ranked and selected independent of its neighbours or location, then we can have a meaningful evaluation of the summarization system.

Our experiments showed that the *Roget's SemDist* ranker performed best when evaluated with Macro-Average Precision. Although the SCU scores from our evaluation in Section 3.2 did not show *Roget's SemDist* to have much advantage over *tf.isf* in terms of unique SCU weight, we found that it performed much better in terms of the ratio of positive to negative sentences. We also found that for sentence ranking the 1911 version of *Roget's* performed just as well as the 1987 version, which is unusual, because generally the 1987 version works better on problems using semantic relatedness [10].

Acknowledgments

NSERC and the University of Ottawa support this work. Thanks to Anna Kazantseva for many useful comments on the paper.

References

1. Nenkova, A., Passonneau, R.J.: Evaluating content selection in summarization: The pyramid method. In: HLT-NAACL, pp. 145–152 (2004)
2. Nenkova, A., Passonneau, R., McKeown, K.: The pyramid method: Incorporating Human Content Selection Variation in Summarization Evaluation. ACM Trans. Speech Lang. Process. 4 (2007)
3. Copeck, T., Inkpen, D., Kazantseva, A., Kennedy, A., Kipp, D., Nastase, V., Szpakowicz, S.: Leveraging DUC. In: HLT-NAACL 2006 - Document Understanding Workshop, DUC (2006)

4. Nastase, V., Szpakowicz, S.: A Study of Two Graph Algorithms in Topic-Driven Summarization. In: Proc. TextGraphs: 1st Workshop on Graph Based Methods for Natural Language Processing, pp. 29–32 (2006)
5. Fuentes, M., Alfonseca, E., Rodríguez, H.: Support Vector Machines for Query-Focused Summarization Trained and Evaluated on Pyramid Data. In: Proc. 45th Annual Meeting of the ACL, Poster and Demonstration Sessions, pp. 57–60 (2007)
6. Katragadda, R., Pingali, P., Varma, V.: Sentence Position Revisited: a Robust Light-Weight Update Summarization 'baseline' Algorithm. In: CLIAWS3 2009: Proc. Third International Workshop on Cross Lingual Information Access, pp. 46–52 (2009)
7. Lin, C.Y., Hovy, E.: Identifying topics by position. In: Proc. Fifth Conference on Applied Natural Language Processing, Morristown, NJ, USA, pp. 283–290. Association for Computational Linguistics (1997)
8. Katragadda, R., Varma, V.: Query-Focused Summaries or Query-Biased Summaries? In: Proc. ACL-IJCNLP 2009 Conference Short Papers, pp. 105–108 (2009)
9. Jarmasz, M., Szpakowicz, S.: Roget's Thesaurus and Semantic Similarity. In: Recent Advances in Natural Language Processing III. Selected papers from RANLP 2003. CILT, vol. 260, pp. 111–120. John Benjamins, Amsterdam (2004)
10. Kennedy, A., Szpakowicz, S.: Evaluating Roget's Thesauri. In: Proc. ACL 2008: HLT, Association for Computational Linguistics, pp. 416–424 (2008)
11. Jarmasz, M., Szpakowicz, S.: Not As Easy As It Seems: Automating the Construction of Lexical Chains Using Roget's Thesaurus. In: Proc. 16th Canadian Conference on Artificial Intelligence (AI 2003), Halifax, Canada, pp. 544–549 (2003)
12. Radev, D.R., Jing, H., Styś, M., Tam, D.: Centroid-Based Summarization of Multiple Documents. Inf. Process. Manage. 40, 919–938 (2004)
13. Zhu, M.: Recall, Precision and Average Precision. Technical Report 09, Department of Statistics & Actuarial Science, University of Waterloo (2004)
14. Carbonell, J., Goldstein, J.: The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. In: Research and Development in Information Retrieval, pp. 335–336 (1998)