Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference

A Computationally Efficient System for High-Performance Multi-Document Summarization

Sean Sovine, Hyoil Han
Marshall University
One John Marshall Drive
Huntington, WV 25755

Abstract

We propose and develop a simple and efficient algorithm for generating extractive multi-document summaries and show that this algorithm exhibits state-of-the-art or near state-of-the-art performance on two Document Understanding Conference datasets and two Text Analysis Conference datasets. Our results show that algorithms using simple features and computationally efficient methods are competitive with much more complex methods for multi-document summarization (MDS). Given these findings, we believe that our summarization algorithm can be used as a baseline in future MDS evaluations. Further, evidence shows that our system is near the upper limit of performance for extractive MDS.

Introduction

Due to modern information technology, text information is now produced and disseminated at a very rapid pace. This situation leads to information overload when users are faced with a large body of text documents relevant to an information need and no efficient and effective way to locate specific information among those documents. For example, imagine a scenario in which a user has a collection of digital news articles relevant to a current event and needs to rapidly generate a summary of the essential information from those articles about the event.

Multi-document summarization (MDS) is one attempt to solve the problem of information overload, and consists of the study of automated techniques for extracting key information from a set of documents with a common topic and using that information to form a concise summary of the documents in the set. In extractive MDS, key sentences from the source document set are extracted to form the summary. Currently, most successful MDS systems are extractive in nature. The alternative, abstractive summarization is a much more difficult task entailing the generation of natural language. We propose an algorithm for extractive MDS that is computationally efficient and also exhibits near state-of-the-art performance on several data sets.

Most MDS algorithms can be divided into three fundamental parts: pre- and post-processing, sentence ranking, and summary generation. *Pre- and post-processing* entails all the basic processing tasks to prepare document text for summarization, such as sentence segmentation, conversion to vector format, and stop-word removal, and any processing that is done after summary sentences are selected, such as ordering the sentences for readability. In *sentence ranking*, document sentences are given scores that reflect the importance of each sentence for inclusion in a summary. In *summary generation* a set of sentences is selected for the summary that simultaneously maximizes total score while minimizing redundancy.

Many existing summarization systems use fairly elaborate, computationally intensive methods for generating automatic summaries. For example, while the system *CLASSY* uses a fairly straightforward method for scoring sentences, it also utilizes a three-stage procedure for selecting sentences that involves a matrix singular value decomposition (SVD), a pivoted-QR matrix factorization, and solving an integer linear program (ILP) knapsack problem (Conroy et al. 2010). The system submitted to TAC 2010 by researchers from IIIT, Hyderabad utilizes support vector regression to combine features as well as a query focusing technique using the probabilistic hyperspace analogue to language model (Varma et al. 2010). The TAC 2010 system submitted by the Joint Research Centre utilizes sophisticated information extraction tools such as geo-tagging and entity disambiguation with a latent semantic analysis (LSA) -based summarization method (Steinberger, Tanev, and Steinberger 2010). The techniques used in these systems and many others involve considerable conceptual and computational complexity.

This paper makes two important contributions. First, we develop a conceptually simple and computationally efficient system and show that its performance is competitive with the top systems at the DUC and TAC conferences [1]. Second, we compare the effectiveness of three methods for generating summaries from a ranked list of document sentences. Because our system performs competitively with top systems across DUC and TAC test data sets from a number of years, we believe that our system would be effective as a baseline for evaluating the performance of future summarization efforts.

[1] Document Understanding Conference and Text Analysis Conference. See http://duc.nist.gov and www.nist.gov/tac.

The rest of the paper is organized as follows. In the next section, we provide background information on using simple features in MDS and summary evaluation. In the section **Sentence Ranking**, we describe how our system estimates the importance of sentences for inclusion in a summary. In **Summary Generation**, we discuss the techniques we utilize to select sentences in a way that attempts to maximize the informative content of the summary while reducing redundancy. In **Evaluation**, we discuss the design of the experiment we utilized to test the effectiveness of our system against existing state-of-the-art systems, and in **Conclusions** we discuss our findings and future work.

Background

Related Work

Previous work has shown that simple algorithms using combinations of simple features for ranking sentences can yield near state-of-the-art performance in MDS (Bysani, Reddy, and Varma 2009). Nenkova and Vanderwende showed that word frequency alone is a powerful feature in MDS, and developed the *SumBasic* system using only word frequency (Nenkova and Vanderwende 2005). The *SumBasic+* system builds on the foundation of SumBasic, also using word frequency statistics, and adds improvments in redundancy removal and pre- and post-processing (Darling 2010).

General MDS Process

All extractive MDS systems contain three essential components: sentence ranking, summary generation, and pre- and post-processing. In extractive MDS, since summaries are formed by extracting document sentences, the fundamental question of extractive MDS is which sentences are most important for inclusion in the summary. In general, most systems approach this decision in two stages. The first stage, sentence ranking, determines the importance of each sentence based on various features of the individual sentence. In the second stage, summary generation, not only individual qualities of sentences are considered, but also the qualities of sentences in combination, as a candidate summary. The most salient issue in summary generation is redundancy, which occurs when two sentences both contain significant, but highly similar content.

Figure 1 outlines the architecture of our summarization system. The stages of our system shown inside the box with the broken outline in Figure 1 are pre-processing steps to prepare sentences for sentence ranking and summary generation.

Summary Evaluation

The National Institute of Standards and Technology (NIST) has sponsored an annual text summarization competition since 2000: the Document Understanding Conference (DUC), which later became a part of the Text Analysis Conference (TAC). In order for progress to be made in the summarization field, it is of vital importance that there is an objective, scientific methodology for measuring summary quality. The TAC/DUC conferences facilitated the development of such a methodology.

Two specific evaluation systems have been used in the DUC/TAC conferences: ROUGE, which is a fast, automated method based on n-gram matching between machine summaries and reference summaries, and Pyramid, a systematic human evaluation system based on locating units of content that overlap between machine and human summaries (Lin 2004)(Nenkova, Passonneau, and McKeown 2007). While Pyramid offers a more rigorous evaluation than ROUGE, the human effort required by Pyramid makes it only practical for large-scale research efforts and conferences. Like most MDS researchers, we use ROUGE to evaluate our MDS system against state-of-the-art systems from recent years.

While many of the evaluation tasks we use to evaluate our system feature a query-focusing MDS element, our system ignores the query portion of the task, in the interest of exploring techniques for generalized summarization. It seems fairly clear that utilizing the extra information held in a query could only improve our system's performance. Thus, it should also be noted that our test results reflect a comparison of our system, which does not utilize query information, to systems which did use query information, meaning our system should be at a slight disadvantage.

Simple Features in MDS

In work of Bysani, Reddy, and Varma, it was shown that *Document Frequency Score* (DFS), which essentially measures the average number of documents a word from a particular sentence occurs in, is a strong indicator of sentence relevance, when used in combination with other simple features (Bysani, Reddy, and Varma 2009). The DFS score rewards sentences that contain words that appear in many documents in the input set and penalizes sentences containing words whose occurrence is concentrated in a small number of documents.

Erkan and Radev introduced the idea of using the *centrality* of a sentence to determine its importance for inclusion in a summary (Erkan and Radev 2004). While these authors utilize more complex graph-based methods, we utilize a simple technique to determine sentence centrality, in which the centrality of a given sentence is the average value of that sentence's cosine similarity score when taken with each other sentence in the input document set. We call this technique *cosine centrality*.

We utilize both features DFS and cosine centrality for scoring sentences, resulting in a sentence scoring system which is computionally efficient.

Sentence Ranking

We utilize two features to estimate the importance of a sentence for inclusion in the summary: document frequency score (DFS) and cosine centrality. Before calculating these feature scores, we convert each sentence to a *bag-of-words* representation, that is, each sentence is represented as a vector in V dimensional space, where V is the number of unique word tokens in the vocabulary of the document set. The nth component of a bag-of-words vector s_k for a sentence k corresponds to the number of instances of word n in sentence k. For example, consider the following sen-

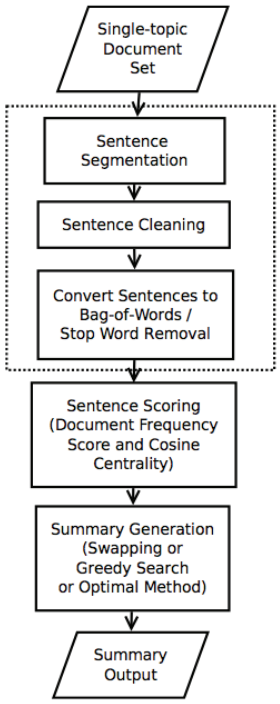

Figure 1: System Architecture

tence from a hypothetical document set with vocabulary $\langle John, went, store, home\rangle$:

John went to the store, then went home.

The vector for this sentence would be $\langle 1, 2, 1, 1\rangle$. Notice that the words 'the', 'then', and 'to' are omitted from the vocabulary. This is because these words are considered *stop words*, which are frequently occuring words that add no significant information about the meaning of the text.

Document frequency score is the average proportion of documents in the input set that a word from a particular sentence occurs in, and is calculated using the following formula:

$$DFS(s_i) = \frac{1}{n}\sum_{j=1}^{n} docFreq(w_{ij}),$$

where n is the number of words in sentence i, w_{ij} is the vocabulary index of the jth word in sentence i, and $docFreq(w_{ij})$ is the proportion of documents in the input set containing the term w_{ij}.

Cosine centrality for sentence i is the average cosine similarity of sentence i to other sentences in the document set and is calculated using the following formula:

$$CosCent(s_i) = \frac{1}{N-1}\sum_{i\neq j\in D}\frac{s_i \cdot s_j}{||s_i||||s_j||},$$

where s_i is the vector corresponding to the ith sentence, D is the set of indices of sentences in the input document set and $N = |D|$.

We combine these two sentence features using a simple linear combination as follows:

$$Score(s_i) = \alpha \cdot DFS(s_i) + (1-\alpha)\cdot CosCent(s_i). \quad (1)$$

We set the value of α manually, based on empirical data. Once the scores for all input set sentences are calculated, the set of all sentences are sorted by score, and the sentence scores are used by the summary generation algorithms to select sentences for inclusion in the summary.

Summary Generation

For summary generation, we consider three different approaches: swapping, greedy search, and optimal. Each of these methods is conceptually simple and computationally efficient relative to the size of the input data set. The optimal method is applied to only to the TAC data sets, which contain 10 documents per set. The input for each of these algorithms is the set of input document sentences, ordered by senence score, which calculated using equation (1).

Swapping

The motivation for the swapping summary generation technique begins with the idea of building a summary by selecting the highest ranked sentences one-at-a-time, while avoiding selecting sentences that are overly similar to already selected sentences. Using this algorithm, a situation can occur in which an already selected sentence is entirely contained in the next sentence on the selection list, but the next sentence contains enough new and important information to merit being added to the summary even so. In this situation it would be desirable to remove the already selected sentence before adding the next sentence.

The swapping method for summary generation utilizes the following algorithm. In this algorithm, S is the score-ranked list of document sentences, C is the ordered set of sentences in the candidate summary, and $S(i)$ is the ith sentence in the ranked sentence list. The function **overlap**(s,t) calculates the proportion of the words contained in sentence s that are also contained in sentence t. The following formula describes the **overlap** function, where $wordSet(x)$ is the set of distinct words in sentence x:

$$overlap(s,t) = \frac{|wordSet(s)\cap wordSet(t)|}{|wordSet(s)|}.$$

Notice that this formula is not symmetric, that is, **overlap**$(s,t) \neq$ **overlap**(t,s), in general.

The function **swap**(C,i,t) removes the ith sentence from candidate C and inserts sentence t in its place, and the function **add**(C,s) adds sentence s to the end of the list C. This algorithm will swap a sentence s_j from the candidate summary with the next sentence $S(i)$ from the ranked sentence list only if **overlap**$(S(i), s_j)$ is greater than or equal to a threshold parameter, β, and **overlap**$(s_j, S(i))$ exceeds **overlap**$(S(i), s_j)$. The threshold parameter β is set based on experimental evidence and has an effect on the diversity of information in the final summary. If **overlap**$(S(i), s_j)$ is less than β, then $S(i)$ is added to C.

Summaries generated by the swapping method generally slightly exceed the target summary length, as can be inferred

from an examination of the swapping algorithm in Figure 2. When a summary exceeds the target summary length, the excess words are truncated from the end of the last sentence in the summary. In the swapping algorithm in Figure 2, the value *sumLength* is the target summary length, in words, and *numWords* is the number of words in the candidate summary C.

```
function swapping(sumLength, S, β)
numWords := 0
i := 0
while (numWords < sumLength)
  maxOverlap := 0
  foreach s_j ∈ C
    if (overlap(S(i), s_j) >
        maxOverlap)
      maxOverlap := overlap(S(i), s_j)
    endif
    if (overlap(S(i), s_j) >= β &&
        overlap(S(i), s_j) > overlap(s_j, S(i)))
      swap(C, j, S(i))
      numWords += length(S(i)) -
          length(s_j)
      break
    endif
  endfor
  if (maxOverlap < β)
    add(C, S(i))
    numWords += length(S(i))
  endif
  i++
endwhile
return C
```

Figure 2: Swapping algorithm: S is the score-ranked list of document sentences, C is the candidate summary, and β is the swapping threshold.

Greedy Search

The greedy search algorithm also begins with the idea of building a summary by selecting the highest ranked sentences one-at-a-time. However, in the greedy search algorithm, after adding a sentence to the candidate summary, that sentence is removed from the score-ranked sentence list, and the remaining sentences are re-scored and re-ranked based on a combination of their original scores and redundancy with the already selected sentences. The motivation for the greedy search algorithm is to build the summary by adding sentences one-at-a-time, while calculating the next optimal sentence after each sentence is selected, based on which sentences have already been selected. This algorithm, by design will only select a sentence to add to the candidate summary if adding that sentence will not cause the summary to exceed the target summary length.

New sentence scores incorporating redundancy with already selected sentences are calculated using the following formula:

$$Score_{new}(s_i) = \alpha \cdot Score(s_i) + (1-\alpha) \cdot CosSim(s_i, C),$$

where C is the vector formed by adding all of the sentence vectors in the candidate summary together, $CosSim(s_i, C)$ is the cosine similarity of vectors s_i and C, and $Score(s_i)$ is the original score of sentence s_i, from equation (1). The parameter α was set empirically.

Optimal

The optimal summary generation method is a restricted exhaustive search method. The summaries generated by this method do not exceed the target length. The optimal method proceeds in four steps, as follows. In this description, S_k is the set containing the top $k\%$ of sentences, based on the sentence score ranking.

1. Select top $k\%$ of sentences, based on score and add to S_k
2. Generate all possible summaries containing sentences from S_k not exceeding target summary length.
3. Calculate score for each summary taking into account length, sentence scores, and redundancy penalty.
4. Select highest scoring summary.

In general, exhaustive search is an inefficient search strategy. However, by considering only the top $k\%$ of sentences, this method becomes feasible for generating summaries.

The score for candidate summary C is calculated using the following formula. In this formula, $v(D)$ represents the document set bag-of-words vector, $v(C)$ is the vector formed by adding all the sentence bag-of-words vectors in C together, and s_i is the ith sentence vector in C:

$$Score_{adjusted}(C) = \frac{wordCount(C)}{targetLength} \cdot Score(C),$$

where

$$Score(C) = \alpha \cdot CosSim(v(C), v(D)) + (1-\alpha)W(C),$$

and the inverse redundancy score $W(C)$ is calculated using

$$W(C) = 1 - \frac{1}{|C|} \sum_{s_i \in C} CosSim(s_i, v(C) - s_i).$$

The score for candidate summary C is a linear combination of the cosine similarity of the summary to the document set, in the bag-of-words representation, and the inverse of the internal redundancy within the summary. In the final score, the score of summary C is penalized if C is shorter than the target summary length.

Evaluation

To evaluate our MDS system, we utilized the evaluation data from DUC 2005-2007 and from TAC 2008-2010. As our system has a number of parameters to be optimized, we required a set of training data. In order to make a fair comparison to the DUC/TAC systems, when evaluating a system using data from a particular year, we only used data from years prior in optimizing parameters. For example, for the 2006 test data, we only used the 2005 data and 2005 reference summaries for training.

Each DUC or TAC test data set contains about 45 document sets, where each document set contains a number

of documents focused on a single topic. The DUC/TAC summarization tasks involve generating a summary for each of the document sets in the test data set. The DUC task was to generate a 250-word summary, while the TAC task was to generate a 100-word summary. In 2005, the number of documents per set ranged from 25 to 50, and in 2006 and 2007, each document set contained 25 documents. The TAC document sets all contained 10 documents each. Since the DUC and TAC tasks respectively involved generating different length summaries from different size input document sets, we believe that using DUC data to train the system for tests on TAC data would not be as effective. Experimental results verified this assumption. The test we performed are as shown in Table 1:

Training Set	Test Set	Test Name
DUC 2005	DUC 2006	D06-05
DUC 2005	DUC 2007	D07-05
DUC 2006	DUC 2007	D07-06
TAC 2008	TAC 2009	T09-08
TAC 2008	TAC 2010	T10-08
TAC 2009	TAC 2010	T10-09

Table 1: Data Sets and Naming Convention for Tests

Results

The following two tables summarize the results of our evaluation. The optimal method was only tested on the TAC data sets, as the larger document sets yielded an impractical number of candidate summaries. In the DUC and TAC conferences, a team was allowed to submit multiple runs. These runs were typically different versions of the same system. For our ranking, we consider the number of teams overall and the number of teams whose systems scored higher than our system, rather than the total number of runs and the higher-ranking runs. For example, in the first row and fourth column of Table 3, the number '$2/24$' means our system was outscored by only one other team at TAC 2009, and 24 teams submitted systems to TAC 2009. We used the ROUGE-2 precision metric as our primary metric for evaluation and training. The scores listed in Table 2 and Table 3 are ROUGE-2 scores.

Test	Generation Method	Score	Rank
D06-05	Swapping	0.09175	2/34
D06-05	Greedy Search	0.08861	5/34
D07-05	Swapping	0.11007	5/26
D07-05	Greedy Search	0.10758	7/26
D07-06	Swapping	0.10874	6/26
D07-06	Greedy Search	0.10678	9/26

Table 2: Results on DUC data sets.

Test	Generation Method	Score	Rank
T09-08	Swapping	0.10849	2/24*
T09-08	Greedy Search	0.10790	2/24
T09-08	Optimal	0.10173	6/24
T10-08	Swapping	0.09448	2/24
T10-08	Greedy Search	0.09687	1/25†
T10-08	Optimal	0.09786	1/25†
T10-09	Swapping	0.09276	2/25
T10-09	Greedy Search	0.09527	3/25
T10-09	Optimal	0.09404	2/25

Table 3: Results on TAC data sets.

As can be seen from our results in Table 2 and Table 3, our system performed competitively across all test sets. In particular, our system scored significantly higher than the highest-scoring system on the 2010 data set. Especially, our greedy search and optimal methods outperform all other systems (as shown in Table 3 with "†"). Our test results also show that the swapping summary generation method performed best overall among our summary generation methods on the DUC data sets (see Figure 3), and the greedy search and optimal methods performed best overall among our summary generation methods on the TAC data sets (see Figure 4). The swapping and greedy search methods performed consistently across all data sets suggesting that these methods are fairly robust when applied to similar data sets, but the optimal method was less consistent.

The TAC 2009 data set included a human-generated extractive summary, which was outperformed by only one system (Genest, Lapalme, and Yous-Monod 2009). This manual extract achieved a ROUGE-2 score of 0.10991. In fact, our system's performance using the swapping method was very close to the score of the manual extract, that is 0.10849 (as shown in Table 3 with "*"). Though more research needs to be done to support this conclusion, the human generated extract should provide an estimate of the upper limit of the performance of automatic extractive MDS systems. If this is the case, then the evaluation results indicate that automated extractive MDS systems are approaching the upper limit of performance for extractive summarization. Further, our system using simple features and a simple algorithm performs near this hypothetical upper limit.

Conclusion

We proposed and developed a conceptually simple and computationally efficient method to perform multi-document summarization. Our system, despite its simplicity, performs competitively with top systems from DUC 2006 and 2007 and TAC 2009 and 2010, many of which involve considerable computational complexity.

Our system uses a linear combination of two features for ranking sentences. These features are document frequency score (DFS), which computes the average proportion of documents in the input set that a word from a sentence appears in, and cosine centrality, which is the average cosine similarity of a sentence to all other sentences in the input set.

For sentence selection, we test three different methods:

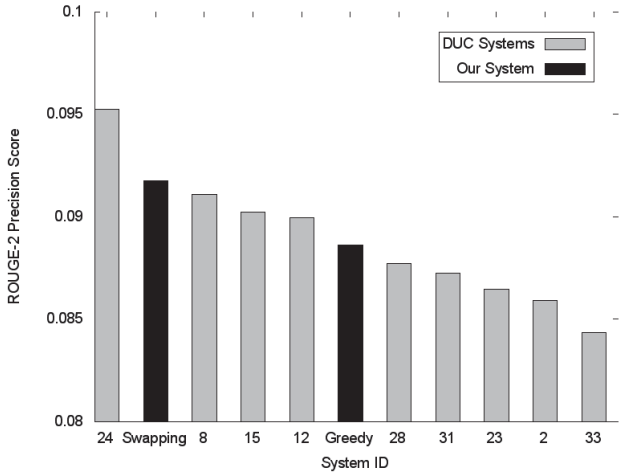

Figure 3: DUC 2006 System Scores

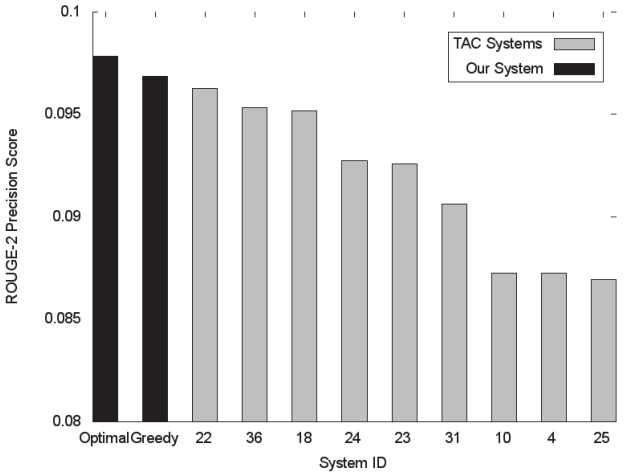

Figure 4: TAC 2010 System Scores

swapping, greedy search, and optimal. Swapping involves swapping out a previously selected sentence with a more informative sentence that contains the same content. Greedy search involves finding the best sentence to add to the summary given the score of the sentence and similarity to already selected sentences. Optimal is essentially a restricted exhaustive search. Optimal is only practical for application to the TAC data sets, which contain a smaller number of documents per set than the DUC data sets. All three methods proved to be effective and competitive, but swapping and greedy search exhibit the best performance.

We have shown that our computationally efficient system can perform competitively with much more complex systems, and that our system displays robust performance across data sets, given the same summarization task. We believe that this implies that our system can be used as a baseline for evaluating future MDS efforts. Also, our system exhibited performance similar to that of a manually generated extractive summary. This shows that the performance of our system and other top systems is near the theoretical limit for extractive summarization.

Future Work

In future efforts, we plan to investigate methods for incorporating query-focusing information into our summarization system. We also hope to incrementally incorporate semantic information obtained from natural language parsing (NLP) techniques into our system, to investigate the gains that utilizing this information may provide. We hope to incorporate these techniques while preserving the essential efficiency of our system.

Acknowledgments

The work described in this paper was partially funded by NASA West Virginia EPSCoR.

References

Bysani, P.; Reddy, V.; and Varma, V. 2009. Modeling novelty and feature combination using support vector regression for update summarization. *Proceedings of The 7th International Conference on Natural Language Processing (ICON 2009)*.

Conroy, J.; Schlesinger, J.; Rankel, P.; and O'Leary, D. 2010. Guiding classy toward more representative summaries. *Proceedings of Text Analysis Conference 2010*.

Darling, W. 2010. Multi-document summarization from first principles. *Proceedings of Text Analysis Conference 2010*.

Erkan, G., and Radev, D. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research* (22).

Genest, P.; Lapalme, G.; and Yous-Monod, M. 2009. Hextac: the creation of a manual extractive run. *Proceedings of Text Analysis Conference 2009*.

Lin, C. 2004. Rouge: A package for automatic evaluation of summaries. *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*.

Nenkova, A., and Vanderwende, L. 2005. The impact of frequency on summarization. Technical report, Microsoft Research.

Nenkova, A.; Passonneau, R.; and McKeown, K. 2007. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Transactions on Speech and Langauge Processing* 4(2).

Steinberger, J.; Tanev, H.; and Steinberger, R. 2010. Jrc's participation in the guided summarization task at tac 2010. *Proceedings of Text Analysis Conference 2010*.

Varma, V.; Bysani, P.; Reddy, K.; Reddy, V.; Kovelamudi, S.; Vaddepally, S.; Nanduri, R.; Kumar, K.; Santosh, G.; and Pingali, P. 2010. Iiit hyderabad in guided summarization and knowledge base population. *Proceedings of Text Analysis Conference 2010*.