Multi-document Text Summarization

Tracy Rohlin
University of Washington
tmrohlin@gmail.com

Karen Kincy
University of Washington
kincyk@uw.edu

Travis Nguyen
University of Washington
travin@uw.edu

Abstract

In this paper, we provide a description of our multi-document summarization system. The documents are parsed using several modules, the sentences of the documents are scored, and the scores of the sentences are used in conjunction with a pre-set word count threshold to produce summaries that are then analyzed by the ROUGE evaluation toolkit.

1 Introduction

Multi-document summarization, a burgeoning field in natural language processing, collects multiple documents and attempts to produce a concise and readable summary by either extractive or abstractive means. Many approaches have been applied to the problem, including graph models, rule-based methods using hidden Markov models, and statistical methods. One of the seminal approaches in multi-document summarization has been MEAD. This centroid-based system was first described by Radev and other researchers in Radev, Jing, and Budzikowska (2000) and Radev, Jing, Stys, and Tam (2004).

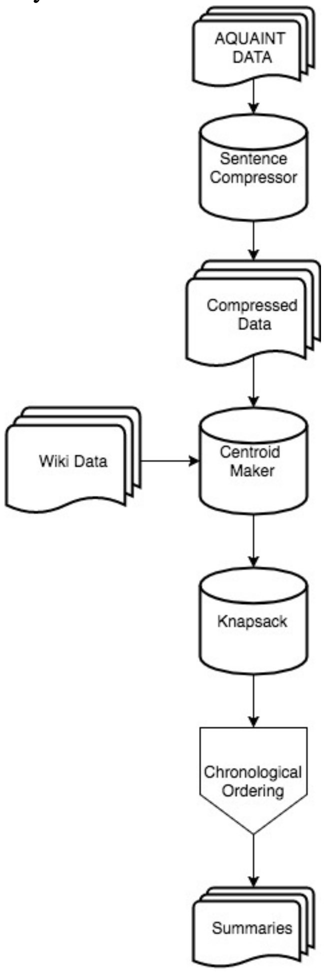
We applied centroid-based summarization to the AQUAINT data set provided by the National Institute of Standards and Technology (NIST), which had been processed using a sentence compression algorithm similar to that described by Zajic, Dorr, Lin, and Schwartz (2007). After content selection, the sentences are ordered chronologically according to an algorithm described by Bollelaga, Okazaki, and Ishizuka (2011). The following sections describe our approach in detail: the general architecture, a detailed description of components, a discussion of results (with ROUGE scores), and finally an error analysis.

2 System Overview

The system is comprised of three major parts: a Standard General Markup Language (SGML) parser that links the document IDs found in the guided summary file to the document SGML files in the corpus folders, a centroid-based algorithm that scores the sentences within each cluster, and a knapsack algorithm that chooses the highest scoring set of sentences constrained by the word count limit of the summary.

2.1 System Architecture

The following graph displays the architecture of our system:



3 Approach

3.1 Content Preprocessing and Realization

The SGML parser uses Python's subprocess library to grep (and in the case of the eval set, zgrep) for each document ID found in the guided summary file. Once found, the exact file is read and parsed using BeautifulSoup and the Natural Language Toolkit (NLTK). This was done to avoid loading all 3,000 files in the corpus folder into memory before linking the document IDs from the summary files to the document files, thereby reducing the risk of running out of memory. The text of each document is retrieved using BeautifulSoup and pre-processed by removing extraneous tags, duplicate headlines, and the document IDs themselves from the text.

This text is then run through a sentence compression algorithm inspired by Zajic, et al. (2007). Prior to compression, regular expressions remove any lingering detritus from the input corpora; this includes datelines, author attributions, editor's notes, and other metadata. We also chose to remove quotations from the text, since they had a negative effect on summary readability. During compression, temporal expressions, adjuncts, and the modals "can" and "have" are removed from the text. The temporal expressions are removed by checking the current tokenized word against a list of day/month words as well as "tonight", "tomorrow", etc.. The previous and following token are also removed if the word is a temporal marker as in "last" for "last Wednesday." Further temporal expressions are removed by part-of-speech tagging, then removing all adverbials except a subset that include directionals such as "up" and "down" and adverbials that change the intent of the sentence (e.g., "nearly", "allegedly").

Further compression as described by Zajic, et al. (2007) was left unrealized as we observed a severe effect on readability. As such, our algorithm does not remove conjuncts, SBARs, determiners, or prepositonal phrases.

The compressed text is saved as a JSON object, where the keys are the topic IDs and the values consist of the tokenized documents along with the document headlines. This cached out JSON file is loaded into the next program in the pipeline for scoring and finally summarization.

3.2 Content Selection

After pre-processing and sentence compression, the centroid-based algorithm loads the JSON file into memory. The module parses the processed text to initialize a list of Cluster instances. Unlike the MEAD algorithm outlined by Radev et al. (2004), our system does not automatically perform any clustering; since each cluster produces its own summary, we defined a cluster as a set of documents with the same topic ID. This allows our system to produce only one summary, avoiding multiple summaries per topic.

Each Cluster object contains a list of instances of the Sentence class, which encapsulates all the data necessary to rank a sentence. The algorithm assigns five scores to each sentence: (1) centroid score, (2) position score, (3) first sentence overlap score, (4) topic focus score, and (5) Wikipedia score. In addition, a redundancy penalty is subtracted from the sum of these scores. More detail about the redundancy penalty follows after a discussion of the scoring mechanisms.

The centroid-based algorithm calculates TF*IDF for each of the terms in a cluster, using a background corpus for inverse document frequency (IDF). To optimize the code, the IDF scores for every term in the background corpus were calculated once and cached out in a JSON file. When testing the choice of background corpora, a subset of the Tagged and Cleaned Wikipedia corpus produced the best ROUGE scores. Due to the large size of the Wikipedia corpus (approximately 67 GB) we processed the first 50,000 documents in the corpus and saved the IDF scores for each term seen in this subset. 50,000 documents more than doubles our previous choice, the Reuters-21578 corpus loaded from NLTK, which has only 21,578 documents. We theorize this increase allows for a more accurate representation of relevancy, as modeled by TF*IDF for each term.

Term frequency is defined as the average frequency of a term across the documents in the cluster. Rather than using the TF*IDF threshold proposed by Radev, et al., centroid size has been parameterized as an argument in the centroid algorithm. A centroid size of 100, for example, saves the top 100 terms for the cluster, ranked by descending TF*IDF scores. During optimization, we found a centroid size of 600 produced the best ROUGE scores for both the development and evaluation sets.

While building a centroid, the algorithm pre-processes the text by tokenizing each sentence within the cluster, lowercasing every token, and removing punctuation-only tokens. A list of cleaned tokens is stored within every Sentence instance, allowing the centroid score for this sentence to be easily calculated. For each token in the sentence, if the token appears in the centroid for this cluster, the token's TF*IDF score is added to the total centroid score for the sentence. This process ignores terms not appearing in the centroid, giving better scores to sentences with relevant terms.

Second, the system calculates the position score for every Sentence instance. The formula from Radev et. al. (2004) prefers first sentences, since the position score of the first sentence in every document equals the highest centroid score for that cluster. Every sentence after the first sentence gets penalized with the following formula, where *n* is the position of the sentence within the document and $c_(max)$ is the value of the highest centroid score for the cluster:

$$P_i = ((n - i + 1)/n) * C_(max)$$

Third, the system initializes the first sentence overlap score for each sentence, defined as the dot product between vectorized sentences. As a form of optimization, rather than vectorizing the text, every Sentence instance contains a dictionary of (word, count) pairs; the overlap between sentences can be quickly calculated by finding the intersection of keys between dictionaries. For every word shared by both sentences, the counts are multiplied and added to the first sentence overlap score.

Unlike MEAD, our system incorporates a fourth score for topic focus. This score is calculated by training a continuous bag-of-words (CBOW) model on the background corpus using Word2Vec. For every token in a candidate sentence, that token is compared to each term in the topic string (e.g., "Giant Panda"). Rather than an exact string match, the CBOW model compares the embeddings for these terms, determining if these words were seen in similar contexts during training. Hand-tuning the similarity threshold, we found *sim(token, topic_word)* $>= 0.75$ produced the best results. If two terms meet this similarity requirement, the topic focus score for the candidate sentence gets incremented by a bonus point. Lastly, the total topic focus score for each sentence is multiplied by the topic focus weight.

Finally, we added a fifth score. The Wikipedia score extends the topic focus score through query expansion. The topic strings were extracted from the guided summary files, e.g., "Cyclone Sidr" from the evaluation set. Each topic string was used as a query to search for the corresponding article on Wikipedia. Initially, we relied upon human disambiguation to choose the Wikipedia article for each topic, though we later explored a more systematic, automatic approach, as detailed in the Results section. The text from each Wikipedia article is pre-processed to produce stopworded, lowercased tokens. All the terms seen in this article are ranked according to TF*IDF scores. The top-scoring 100 terms for each Wikipedia article form a centroid. The centroids for each topic are cached out as a JSON file, which is then loaded by the centroid-based summarization module.

When calculating the Wikipedia score, every token from the candidate sentence is compared with the top 100 tokens from the Wikipedia article for this topic. Like the topic focus score, exact string matches are abandoned in favor of similarity between embeddings, using the pre-trained CBOW model. A threshold of 0.75 was used for similarity. We trained the CBOW model on the first 50,000 documents in the Wikipedia background corpus, using a vector dimensionality of 100, a context window of 5, a maximum vocabulary size of 25,000, and a minimum count of 2, which ignored words seen less than two times in training. Initially, we set the minimum count to 1, including all words seen during training, though we found this increased the size of the saved CBOW model without improving ROUGE scores. Note that the actual vocabulary of the trained CBOW model contains 12,623 embeddings, as opposed to the maximum possible 25,000 embeddings.

After calculating all five of these scores, each of these scores is multiplied by a corresponding weight. Radev et al. discuss weights for MEAD, though they set the weights equal to 1 and leave tuning for future work. We performed parameter optimization for the weights in our system, using a grid search of ROUGE scores, to determine the best values. The final parameters are discussed in the Results section. As mentioned previously, we added the choice of background corpora as a parameter, and found that Wikipedia produced the best results. Switching to Reuters, for example, reduced ROUGE scores substantially.

Finally, a redundancy penalty is subtracted from the sum of the previous three scores. The original MEAD algorithm calculates redundancy pairwise, comparing the similarity between each sentence and the immediately preceding sentence, ranked by total score in descending order. We found this approach still had too much redundancy within the summaries, and opted for a revised approach. The redundancy penalty for our system is cumulative; that is, the overlap for each sentence is calculated using the intersection of the word types of the current sentence to the word types of all sentences with a higher total score.

$$R_s = 2 \times \frac{\#overlappingwords}{\#wordsinprevioussentence + \#wordsincurrentsentence}$$

After the total score of each sentence is calculated, the sentences are ordered by their total score, in descending order, and only the top N are considered to create the summary. After parameter optimization, we found N = 60 produced the best ROUGE scores.

At this point, we introduced another redundancy reduction measure. First, we represent the tokens in each of the top N sentences as word/count pairs in a dictionary, to compute word overlap between sentences. For each sentence, compare it to every other sentence. If the cosine similarity between a pair of sentences exceeds 0.7, mark the sentence with the lower total score as redundant by adding it to a set. After checking all pairs of sentences, take the set of top N sentences and subtract the redundant set. The resulting disjunction contains no identical sentences or nearly identical sentences, which was a problem in our earlier system.

In order to create the summary, we use the knapsack algorithm, which allows us to select a subset of the top N sentences such that the word count of the summary, equal to the sum of the word counts of the sentences in the subset, is less than or equal to a given threshold. The knapsack algorithm also selects a subset of the top N sentences that has the highest total score, calculated as the summation of the total scores of each sentence in the subset. Per the assignment, the threshold for the summary word count limit is set as 100.

3.3 Information Ordering

Our information ordering algorithm is an adaptation of Bollelaga, et al.'s chronological ordering algorithm. Tie-breaking for two first sentences is done by comparing the date of the sentences. The original algorithm sorted sentences first on publication date of the sentence, then on sentence position. Ordering based on sentence position helps to break ties in the case where two sentences have been extracted from the same document and thus have the same date-time associated with it, or two sentences come from separate documents yet still have the same date-time.

The modified version of the algorithm takes the chronological ordering but first tries to order based on whether the sentence is a first sentence or not. If the sentences being compared are not first sentences, it then will compare based on date-time and sentence position. This was found to have improved readability compared to the original algorithm. This may be due to the fact that newspapers often place the most pertinent and most general information towards the top of the article, with detailed information following after the lede. This also prevents summaries where one sentence with an earlier date-time that contains more detailed information is placed before an older but more general lede sentence. The decision to use a modified version of a chronological ordering instead of the full system proposed by Bollelaga, et al. (2012), was due to the results reported in their paper. The proposed system results in a less than 1% (0.9%) improvement over a pure chronological ordering algorithm. Indeed, in their proposed system, the chronological ordering component is the second highest weighted component (the first being succession) in their final optimized system. Thus, for sake of simplicity and time constraints, we felt confident chronological ordering and sentence position ordering were good approaches to take with regards to organizing the content.

4 Results

Reviewing the ROUGE recall scores, the ROUGE-1 scores are consistently the best for summaries produced by our system. This makes sense, since the TF*IDF metric favors relevant unigrams, influencing both the centroid score and the Wikipedia topic focus score. Our system does not currently take into account the relevancy of bigrams, trigrams, or 4-grams, which could presumably be incorporated to increase the ROUGE-2, ROUGE-3, and ROUGE-4 scores.

4.1 Improved results

Table 1 (next page) displays the ROUGE-1, ROUGE-2, ROUGE-3, and ROUGE-4 recall values, in terms of percentage, compared to our original D3 ROUGE scores. We report the scores resulting from just using Wikipedia for the background corpus and topic focus with no sentence compression, the scores resulting from incorporating data from Wikipedia as well as our current sentence compression, and the scores resulting from the above method with an additional attempt at removing conjuncts by way of NLTK parsing. The best scores arise from the system where only adjuncts, temporals, and some modals are removed, as opposed to the additional conjuncts. The scores increased across the board from D3, around 1-2%, though less so for ROUGE-3 and ROUGE-4.

Table 2 displays our final, un-tuned results on the evaluation set using the system incorporating adjunct and temporal expression removal as well as the Wikipedia background corpus and Wikipedia query expansion:

Scores	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4
Evaluation Set	32.139	9.918	3.795	1.796

Table 2: Evaltest Results

After presenting these results for the final deliverable, we explored refinements to the Wikipedia query expansion. In particular, we aimed to replace any subjective human disambiguation of choosing Wikipedia articles with a systematic, automatic approach. We tested two methods:

- Input the topic string into the Wikipedia search function and choose the first result, or allow Wikipedia to redirect to the corresponding article.
- Input the topic string into a Google search query restricted to Wikipedia, such as "site:wikipedia.org Rain Forest Destruction", and choose the first result.

The first approach (Table 3), choosing the first Wikipedia search result, resulted in 34 new articles and 56 unchanged articles when compared to the Wikipedia articles chosen by human disambiguation. This method decreased the scores for the development set around 1% for all ROUGE scores. For evaltest, the scores for ROUGE-3 and ROUGE-4 increased, while ROUGE-1 and ROUGE-2 decreased:

Scores	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4
Evaluation Set	30.147	9.433	3.811	1.925
Development Set	27.44	7.423	2.698	1.221

Table 3: Wikipedia search

The second approach (Table 4), searching Wikipedia using Google, resulted in only 17 out of 90 Wikipedia articles differing from the human disambiguation, which suggests it chooses more relevant articles. It produced even better ROUGE scores than the first method, with a ROUGE-2 of nearly 10 percent for the evaluation set:

Scores	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4
Evaluation Set	31.9	9.9997	3.93	1.952
Development Set	28.276	7.872	2.945	1.283

Table 4: Google search

Clearly, query expansion using Wikipedia can be automated, and increases ROUGE scores, though we leave the actual script to select the relevant Wikipedia articles for future work due to time constraints.

5 Discussion

Overall, we found that sentence compression, parameter optimization, the choice of Wikipedia as a background corpus (for both calculating TF*IDF and training a CBOW model), and Wikipedia query expansion for topic focus improved our system's ROUGE scores.

Currently, the following parameters produce the highest average ROUGE recall scores:

- Centroid Size = 600
- Top N Sentences = 60
- Centroid Weight = 5
- Position Weight = 1
- First Sentence Weight = 1
- Redunancy Weight = 100
- Topic Focus Weight = 10
- Wikipedia Topic Focus Weight = 200

Note the high weight on the Wikipedia Topic Focus score, which seems to indicate its usefulness in improving ROUGE scores.

Scores	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4
D3	25.363	7.330	2.577	1.001
Wikipedia Corpus & Query Expansion	26.499	7.566	2.768	1.161
Wikipedia & Adjunct & Temporal Exp. Removal	**28.582**	**8.174**	**3.052**	**1.323**
Wikipedia & Conjunct Removal	26.200	7.443	2.559	0.994

Table 1: D4 Improvements for devtest

5.1 Error Analysis

For our error analysis, we studied the summaries produced by our system for the development and evaluation sets. Clearly, our system's output still has issues with readability, particular in terms of coreference resolution and cohesion. Some issues may be due to overaggressive sentence compression, since we focused on optimizing ROUGE scores and did not have the time to implement more improvements to readability.

An example summary with readability issues:

> Authorities believe Columbine students Eric Harris and Dylan Klebold carried out the massacre and killed themselves. 12 Columbine High School students and a teacher were murdered when Eric Harris and Dylan Klebold Columbine students opened fire with at least four guns and dozens of bombs. **Who oversees the county high schools.** Diane Hitchingham of the Jefferson County school system said schools were getting phony bomb threats. **Columbine is about three miles.** Columbine and Chatfield are sports rivals. **Including the two killers died** in a shooting and bombing spree at Columbine High School.

The first sentence of this summary introduces the perpetrators of the Columbine High School massacre, but does not identify "the massacre" explicitly. Additionally, the first sentence uses the shorter mention "Columbine" rather than the full mention "Columbine High School", since we did not implement any coreference resolution or improve the mentions of entities.

Compression produced the truncated phrase, "Columbine is about three miles." Referring to the uncompressed corpora, the correct sentence reads: "Columbine is about three miles away." Clearly, the erroneous sentence resulted from "away" being removed with the majority of adverbials, rather than being saved in the subset with directionals like "up" and "down".

Overaggressive compression also produced, "Who oversees the county high schools." The rest of this sentence is unrecoverable from context; we do not know who this person might be. This truncation is likely due the compression algorithm removing phrases that are two to three words long followed by a comma. While this removes many unnecessary transitional phrases such as "in summary" or "in essence," it also removes sentences that have a person's name in the beginning of the sentence followed by an appositive or descriptive phrase. This error also affects sentences that start with number containing a comma, as seen in the following summary:

> Bangladesh has sent medicines and other relief goods for the tsunami victims in Sri Lanka and the Maldives. The Ilyushin-76 cargo plane headed for Medan the main city on **Indonesia Sumatra island** one of the areas **hardest hit 26 tsunami**. Japan has provided some 500 million US dollar fund for tsunami- affected countries including Indonesia the Maldives and Sri Lanka. **000 people** were killed Indonesia. **000 people** were made homeless after tsunami swept Aceh province in Indonesia on Dec. 26. **Indonesia official earthquake and tsunami death toll** rose 100 people to 94,200.

This summary demonstrates two additional bugs: first, the compression algorithm erroneously removes some possessive endings from words. In the previous example, the incorrect phrases should read "Indonesia's Sumatra island" and "Indonesia's official earthquake and tsnunami death toll". Second, more commas would improve readability: "...headed for Medan, the main city on Indonesia's Sumatra island, one of the areas..." The lack of commas also hurts sentences such as, "Simpson

60 faces an 11th felony charge coercion.", which should read, "Simpson, 60, faces an 11th felony charge, coercion."

Besides sentence compression, increasing the number of sentences considered by the knapsack algorithm seems to have created additional problems. With top N = 60, occasionally the system produces pathological summaries with choppy, vague sentences:

> The hospital was evacuated. The police said two men were arrested. The incident came a after police foiled car bomb attacks in London. The incident is treated as a terrorist attack. A blazing car crashed into the terminal building at Glasgow Airport in Scotland on Saturday. Glasgow Airport was evacuated and closed after a burning car crashed into the main terminal building afternoon. It contained inflammable materials. Britain senior police officer said that the attack at Glasgow airport was linked to events in London where two car bombs were defused on Friday.

The shortest sentences should have been scored poorly by the centroid-based algorithm, since they do not appear to be first sentences extracted from documents, or to contain many terms relevant to the topic. A larger top N considers sentences with lower scores, which generally includes shorter sentences, which allows the knapsack algorithm to choose many short sentences rather than a few longer ones to fit in 100 words. In addition, the information ordering for this summary needs to be investigated, with a potential first sentence buried in the middle.

6 Conclusion

In conclusion, our multi-document summarization system pre-processes documents, compresses them according to Zajic, et al. (2007), scores the sentences within each topic cluster, and uses the knapsack algorithm to create a summary, keeping in mind the highest scoring subset of sentences as well as the summary word count threshold. These sentences are then ordered by using a modified version of Bollelaga, et al.'s algorithm, where first sentences are placed at the top of the summary and the other sentences are ordered chronologically and then by sentence position. The final summaries are then evaluated using the ROUGE evaluation toolkit. The additions to our system do improve ROUGE scores across the board, though there are still improvements needed for future work.

References

Danushka Bollegala, Naoaki Okazaki, & Mitsuru Ishizuka. 2012. *A Preference Learning Approach to Sentence Ordering for Multi-Document Summarization*. Journal of Information Sciences. Pages 78-95.

Dragomir R. Radev, Hongyan Jing, & Malgorzata Budzikowska. 2000. *Centroid-based Summarization of Multiple Documents: Sentence Extraction, Utility-based Evaluation, and User Studies*. NAACL-ANLP-AutoSum '00 Proceedings of the 2000 NAACL-ANLP Workshop on Automatic Summarization. Pages 21-30.

Dragomir R. Radev, Hongyan Jing, Malgorzata Stys, & Daniel Tam. 2004. *Centroid-based Summarization of Multiple Documents*. Information Processing and Management 40. Pages 919-938.

David Zajic, Bonnie Dorr, Jimmy Lin, & Richard Schwartz. 2007. *Multi-candidate Reduction: Sentence Compression as a Tool for Document Summarization Tasks*. Information Processing & Management. Pages 1549-1570.