Deliverable #2: Base End-to-End Summarization System

Paige Finkelstein **Jacob Hoffman** **Wesley Rose** **Joshua Tanner**
University of Washington
Department of Linguistics
{plfink, jrhoff, warose91, jotanner}@uw.edu

Abstract

This report discusses the implementation of our base end-to-end system for performing multi-document automatic summarization. Our system is designed to take topic-oriented groups of documents and to produce a short extractive summary for each group. We describe our system architecture, which includes the four primary components: pre-processing, content selection, information ordering, and content realization. At this juncture, the majority of our summarization efforts have focused on content selection, and we describe in detail the two primary approaches that we have pursued: one that implements a unique take on an n-gram heuristic, and one that builds upon the graph-based lexical similarity comparison approach of LexRank. We also present our ROUGE score results, offer some error analysis of our current system, and briefly discuss some ideas for future improvements.

1 Introduction

We have implemented a multi-document summarization system that is designed to produce short summaries (a maximum of 100 words) given topic-oriented document clusters of articles from the AQUAINT and AQUAINT-2 corpora. For this deliverable, the bulk of our efforts have been focused on implementing the end-to-end system setup and necessary tooling—including modules for extracting and pre-processing the corpora data and integrating the ROUGE evaluation toolkit—and also on experimenting with a robust initial implementation of our content selection approach.

For content selection, we pursued one method that relied on an n-gram heuristic and NER inspired by work such as sumBasic and HeirSum mentioned in Haghighi and Vanderwende (2009) in order to select sentences for salience and a second method that implemented our own take on the graph-based lexical similarity comparison approach of LexRank described in Erkan and Radev (2004). We ultimately chose to use the custom n-gram heuristic method for this deliverable, but we anticipate using strategies from both approaches combined with additional features for future deliverables.

Our current system implementation achieves an average ROUGE-1 recall score of 0.2165 and an average ROUGE-2 recall score of 0.0615.

2 System Overview

2.1 System Architecture

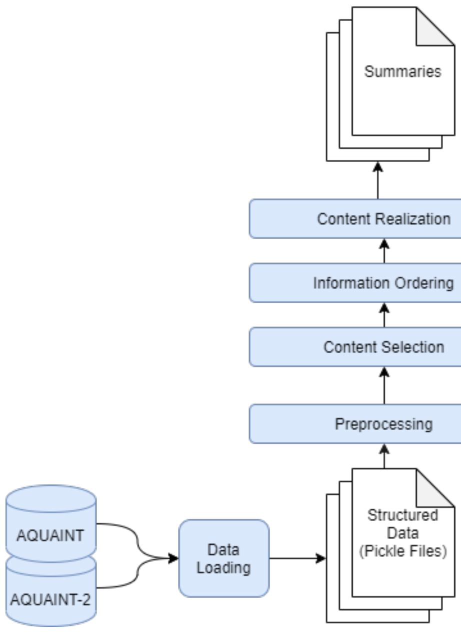

Figure 1: System architecture diagram

As Figure 1 shows, our summarization system is comprised of modules to fetch and load the articles for each topic cluster for the specified corpus, to perform data cleaning and pre-processing, and to perform each of the major components of summarization, namely, content selection, information ordering, and content realization. These modules are each described in detail in the following sections. We have also written a utility module to generate corpus-specific configuration files that are used by the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) toolkit (Lin, 2004) for evaluating our summary outputs.

2.2 Corpus Loading

We wrote abstractions for the AQUAINT and AQUAINT-2 corpora that allow us to easily load files from either of them. For the summarization task files, we first read in the list of requested articles and convert these to a set of *queries* which are then resolved to specific corpus file locations using details such as the year of the article and its journal ID. Once we have resolved all queries to files, we open each required file and retrieve all the necessary articles. The final list of articles corresponding to the summarization task file is serialized and saved into our data directory for easy reloading. We use these same abstractions over the corpora to allow us to easily load all articles from all files in either corpora to compute metrics like TF-IDF.

AQUAINT-2 is loaded with a standard XML parsing library, but because the AQUAINT files are not standard XML, care is taken to make sure these load and parse properly as XML and that we do not drop entities corresponding to various character representations like `AMP;` for ampersands.

2.3 Document Pre-processing

After the corpus has been loaded, the documents have been separated into topic-clusters and each article has been broken into paragraphs, we perform pre-processing to remove unnecessary and noisy artifacts from the documents and to transform them into data structures that will be useful in the summarization modules.

For cleaning up and standardizing the document text, we use a long series of regex commands that have been heavily influenced by data samples in the two corpora. This includes changes such as standardizing quotation marks and removing news article metadata such as *Story Filed By Cox Newspapers*.

For crafting robust data structures to be used in our main summarization pipeline, we use the open source NLP library spaCy (Honnibal and Montani, 2017). spaCy provides support for a wide range of NLP tasks, including part-of-speech tagging, tokenization, named entity recognition, sentence segmentation, etc., and it offers easy integration with pre-trained statistical models and word vectors. In the pre-processing phase, we initialize a spaCy English language model and use it to transform the text for each article, one paragraph at a time, into spaCy `Doc` objects to allow the summarization system modules easy access to built-in spaCy functionality.

2.4 Summarization System

The core portion of our system that does the actual multi-document, extractive summarization work is comprised of three main modules: (1) content selection, (2) information ordering, and (3) content realization. We discuss the current status and implementation of each of these in detail in the following section.

3 Approach

3.1 Content Selection

3.1.1 Heuristic Approach

Currently, our heuristic selection method consists of using n-grams and NER. Using a group of documents within a single topic, we calculate unigram, bigram, and trigram probabilities after removing stop-words and punctuation. The headline of each article is also included when counting n-grams. Then we run through each article one at a time and give each sentence a score. We've included 4 components of the score in our baseline system: (1) unigram score, (2) bigram score, (3) trigram score, and (4) headline named entity overlap score. More concretely, each of the four scoring components is calculated as follows.

1. The sum of the unigram probabilities of each word in the sentence.

2. The sum of the bigram probabilities of each word in the sentence.

3. The sum of the trigram probabilities of each word in the sentence.

4. For each named entity in a headline, E, if an n-gram overlaps with E, then contribute that n-gram's $probability * lambda_N$ to this score.

Additionally, each component has a weight, lambda1-4, that we report in the results section.

Once we have scores for all sentences in a given article, we simply take the N greatest scores, order them by location in the article, and pass them along to the information ordering module for further ordering with respect to the entire article set. We haven't yet solidified how we want to deal with weights and passing them along to later parts of the pipeline for ordering or realization, but we feel this in an area of improvement in the future.

The best average ROUGE-2 recall score we achieved with this method was 0.06145. Surprisingly, we found that including score components #3 and #4 with too high of a weight negatively affects results. We also experimented with a 5th metric which is not included in the baseline due to degrading performance. This metric checked for n-gram overlap with the complete headline text. It is possible that because we are counting the n-grams in the headline towards the n-gram calculations to begin with, these n-grams may be over-considered in our current algorithm. We plan to do more testing in order to investigate how to best incorporate the headline into this approach.

3.1.2 LexRank Approach

We initially intended to use a variation of LexRank (Erkan and Radev, 2004) for our baseline system. While we ultimately chose to use the heuristic approach as our baseline for D2, we still plan for LexRank to be a part of our eventual system and thus felt we should include a section describing our work on it.

Concretely, LexRank did not become part of our D2 baseline for two main reasons: it was unclear whether our unique implementation would finish in time, and perhaps more importantly, the n-gram based heuristic approach was producing better ROUGE results when compared to a prepackaged LexRank implementation that we experimented with for comparison. These comparative scores can be seen in Table 1. We still feel that LexRank could be useful and possibly even competitive with our heuristic in the long run, but will likely require modifications, such as taking the title into account and weighting it more heavily than other sentences.

While we are not yet using it in D2, as part of our effort to build our own unique variation on LexRank, we implemented code to produce word counts by document and consequent IDF scores for the entirety of either corpora, and we look forward to using TF-IDF in LexRank or other aspects of D3.

3.1.3 Long Term Target

Ultimately, our goal is not to choose between our current heuristic approach and LexRank but to integrate all of these (and other methods) into a simple machine learning classifier as described in Hong and Nenkova (2014).

3.2 Information Ordering

For our base implementation of the information ordering strategy, we have taken inspiration from some of the ideas in Bollegala et al. (2012), which outlines a "preference learning" approach to ordering for summarization with multiple documents. Specifically, we incorporate methods that are comparable to two of the "experts" they describe: the *chronological expert* and the *topical-closeness expert*.

For chronological ordering, we follow a very similar paradigm whereby article publication date is given preference in determining order, followed by order in the document. For topical closeness, rather than use this measure to order topically-related sentences together, as Bollegala et al. do, we are using it as a measure to determine redundant sentences and either discard them or move them to the bottom of the queue where they will only be included if there is extra summary space. Like Bollegala et al., we determine topical closeness by comparing cosine similarity between word vectors. Where they use a one-hot encoded vector to represent each sentence, however, we use spaCy to compare an average of the word vectors, which are GloVe vectors trained on Common Crawl. We experimented with removing stop words and lemmatizing verbs and nouns before computing the sentence vectors as Bollegala et al. describe, but found that this actually led to the cosine similarity scores being less aligned with our intuitive human judgments.

3.3 Content Realization

Our content realization module is currently responsible for two primary subtasks:

1. Removing verbiage that is unnecessary for the summary

2. Cutting the summary to 100 words or fewer by selecting sentences provided by previous modules, in order, until the threshold is reached

To remove unnecessary verbiage, we have applied two simple rules inspired by previous work.

As described in Conroy et al. (2006), we have removed sentence-initial conjunctions and adverbs. Following Zajic et al. (2007), we have removed appositives. Our system supports two options for limiting summaries to 100 words or fewer. The default method is to include only full sentences until there are no longer any sentence options that will results in a maximum of 100 words. The second method is simply to truncate the text to 100 words, irrespective of sentence boundaries.

For future deliverables, we plan to apply further methods from existing work while also analyzing our intermediate results and exploring novel approaches. We also plan to add the responsibility of ensuring coherence and cohesion across the summary to this module, which we anticipate being an especially challenging task.

4 Results

Table 1: Comparative Average ROUGE Recall Scores

Method	ROUGE-1 R	ROUGE-2 R
Custom Heuristic	0.21649	0.06145
LexRank	0.24608	0.05528

Table 2: Heuristic Method Average ROUGE Scores

Metric	R	P	F
ROUGE-1	0.21649	0.25296	0.23119
ROUGE-2	0.06145	0.07267	0.06588

For the heuristic method results reported in both Tables 1 and 2, the values used for lambdas 1-4, discussed above, were 0.4, 0.7, 0.05, and 0.05.

As shown in Table 1, our heuristic approach and the LexRank implementation achieved ROUGE-2 R scores of 0.06145 and 0.05528, respectively. Both of these scores are higher than the LEAD baseline R of 0.05376, and the heuristic approach is also higher than the MEAD baseline R of 0.05927. Despite there being a lot more we can do to improve each of our methods, such as better headline inclusion, integrating subtopics, incorporating more information into content ordering, etc., they still achieve reasonable R scores compared to LEAD and MEAD. As such, the heuristic method and LexRank seem to be acceptable candidates for incorporating into our content selection module moving forward.

5 Discussion

Using some existing approaches combined with carefully engineered pre-processing and content selection, we have implemented a competitive baseline. However, we are confident that there is room for improvement in future deliverables. In addition to exploring methods discussed in previous works on document summarization, we plan to tailor solutions specifically to the errors produced by our existing system. Our error analysis can be organized following our summarization system architecture: Problems with pre-processing, content selection, content ordering, and realization.

Our pre-processing module is performing quite well and we haven't noticed any major problems. There are a number of cases where named entities are uncapitalized that we may try to fix, although this is unlikely to affect evaluation. Additionally, we are sure to find minor things as we do a more in-depth analysis and we'll add more pre-processing steps as we go, especially if we end up incorporating additional corpora for training our system.

For content selection, there are a few things that we plan to address. For one, we sometimes see very short sentences that don't contribute any value to the summary. For example, one summary includes the single-word sentence: *Columbine!* To address this, we are considering including sentence length as a feature during content selection, or potentially using a length constraint in the content realization module to inform filtering. We also still need to improve our method for selecting the most relevant content and *only* relevant content, and plan to spend more time analyzing our output summaries in order to determine specific problematic patterns.

For information ordering, we have found some examples of incorrect ordering in our summary outputs but we currently do not have a straightforward fix in mind. As an example, we have a summary beginning with the sentence *The four officers fired 41 shots, hitting diallo 19 times.* We need to find a better way to ensure that we introduce the subject before providing secondary details. One idea is to make sure our summaries begin with a sentence whose subject includes the fullest version a named entity. We are also interested in experimenting with variations on the additional *experts* described in Bollegala et al. (2012) to determine ordering preference.

For content realization, at this point we have only implemented a few simple rules, and there is much room for more work. However, there are already a couple of specific issues to consider. First, we have found a few examples of subordinate clauses that should be removed. Consider the clause starting with *whose* in the following sentence: *Two days earlier, a massacre by two students at columbine high, whose teams are called the rebels, left 15 people dead.* We also need to re-think our approach to removing sentence-initial adverbs and junctions, as this leaves us with some ungrammatical sentences. For example, when we remove *wherever* from the sentence *Wherever Hurricane floyd hits, the federal agency responsible for emergencies says it won't be as tardy as it was after hurricane andrew, seven years ago.* To fix problems like this, we could either use a list of exception words, come up with a more careful heuristic, or simply leave this rule out because it has only delivered a very marginal improvement.

6 Conclusion

In this deliverable we have implemented a complete end-to-end extractive multi-document summarization system and are satisfied with this system as a baseline to build off of in future deliverables. We were somewhat surprised to see the comparative success of our heuristic approach over a LexRank implementation in terms of ROUGE score performance.

In future work we are looking forward both to improving our current content selection method—most likely by combining our heuristic method and a custom LexRank implementation with other features in a machine learning classifier—and to fleshing out more sophisticated approaches for information ordering and content realization.

References

Danushka Bollegala, Naoaki Okazaki, and Mitsuru Ishizuka. 2012. A preference learning approach to sentence ordering for multi-document summarization. *Inf. Sci.*, 217:78–95.

John M Conroy, Judith D Schlesinger, Dianne P O'Leary, and Jade Goldstein. 2006. Back to basics: Classy 2006. In *Proceedings of DUC*, volume 6, page 150.

G. Erkan and D. R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370, Boulder, Colorado. Association for Computational Linguistics.

Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multi-document summarization. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 712–721, Gothenburg, Sweden. Association for Computational Linguistics.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

David Zajic, Bonnie J Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing & Management*, 43(6):1549–1570.