Ling 573 - NLP Systems and Applications

Elena Khasanova, Erica Gardner, Saumya Shah, Sophia Chan and **Vikash Kumar**
Department of Linguistics
University of Washington, Seattle
`{ekhas1, erstgard, saumyahs, schan2, vikas134}@uw.edu`

Abstract

Text summarization is the process of distilling the most important information from a source to produce an abridged version for a particular user and task. In this paper, we discuss a method used for extractive multi-document text summarization. We use topic modelling to find the sentences in a document that are most relevant to a topic. Thus we rank the most relevant words in a topic and uses it to compute a phi score or a word-topic score. The phi score allows us to pick out a sentence that strongly represents the topics in the document. Our end-to-end system produces a ROUGE-2 Average Recall score of 0.06179 on *devtest*, and a score of *0.06676* on *evaltest*. Our system performs slighly better than the baselines.

1 Introduction

Automatic Text Summarization is one of the most challenging and interesting problems in the field of Natural Language Processing (NLP). The demand for automatic text summarization systems is spiking these days thanks to the availability of large amounts of textual data. There are many different paths from a set of documents to a summary.

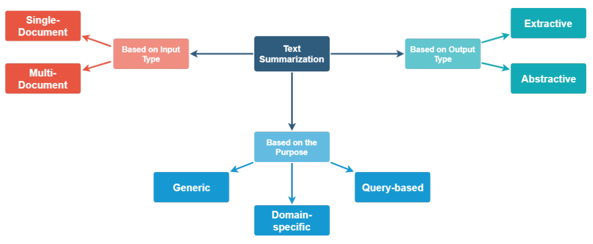

Figure 1: Types of summarization methods

Figure 1 shows the different types of summarization methods based on the type of input, type of output and purpose. We explore the task of extractive multi-document summarization for our project.

For our system, we have explored multidocument extractive summarization methods. Our pipeline follows as having the following stages: content selection, information ordering and content realization. We have experimented with different content selection and information ordering methodologies. Our content selection module uses a topic-modelling and TF-IDF vectors to rank the top sentences in the input documents. The top sentences are passed on to our information ordering module to check for coherence and subsequently the content realization module to remove redundant phrases and improve readability.

2 Related Work

For the different steps in our implementation, we have relied on certain research areas whose implementations have allowed us to improve our model. In the content selection module, we have made use of TF-IDF that is calculated in the preprocessing stage to aid the topic modelling algorithm to choose the most important topics based on the TF-IDF score.

We follow the implementation of Latent Dirichlet Allocation proposed by (Blei et al., 2003). The goal of LDA is to best estimate the topics that could have generated the observed documents. Each document is described as a distribution over a fixed number of topics and each topic is described as a distribution over the known vocabulary. For our purposes, we build an LDA model with three topics and used the most relevant terms of each topics as a measure of word relevance.

For content selection methods, we have also explored the use of graph-based methods such as TextRank (Barrios et al., 2016), which is based on the PageRank algorithm which was used to compute the rank of web pages. Graph-based ranking algorithms are a way for deciding the importance of a vertex within a graph, based on global information recursively drawn from the entire graph. Apart from that we have tested the use of skip-thought vectors (Kiros et al., 2015) for Sent2Vec representations and performing operations on the encoded sentences.

For information ordering, we used the formula in (Barzilay and Lapata, 2008) to calculate coherence scores which is the average cosine similarity across the sentences. To explore semantic coherence we studied the approach proposed by (Foltz et al., 1998). We also used the BERT (Devlin et al., 2018) sentence embeddings to calculate cosine similarity between the sentences.

For content realization, we followed the heuristics suggested in (Conroy et al., 2006) to trim our sentences.

3 System Overview

Our approach highlights a multi-document generic extractive summarization system.

We first get the document sets from the topics using XML readers and organize them to be fed into the preprocessing module. The preprocessing module reads all the documents by topic. The main steps involve stop word removal, lemmatization, sentence segmentation, and tokenization. At this stage we also produce TF-IDF scores for the words in the documents and pass it to the LDA algorithm to choose the top *n* sentences from the topics it generates.

To perform summarization, we use sentence ranking using topic modelling. Topic modelling is an unsupervised method of finding topic clusters within a document and assign a set of words to each topic. Based on the ranking of these words we can find the sentences that contribute most to the topic and hence select content for the summary. Using the word-topic score we find the sentences that are most relevant to summary and is then passed on to the information ordering module.

In the information ordering module, previously, we selected the sentences based on the LDA score from the content selection output and picked the highest ranked sentences to go first. In this deliverable, we have also produced coherence scores using cosine similarity. We also remove the redundant sentence using cosine similarity with sentence embeddings.

For content realization, we experimented with a couple heuristics to get to the 100 word limit such as eliminating sentences shorter than 8 words, removing parenthetical expressions, removing adverbs and gerunds.

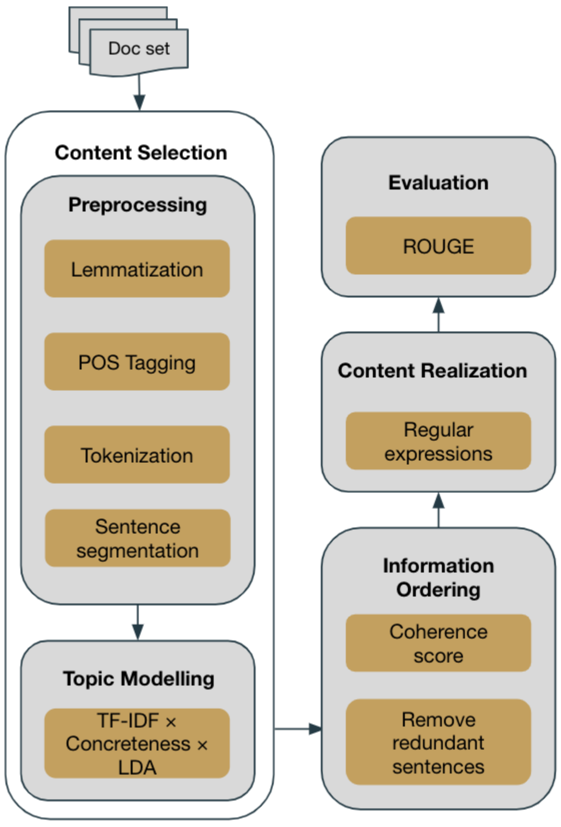

Figure 2: Our proposed system architecture

4 Approach

4.1 Data Extraction

To extract the data from train, test and development files, we use an XML reader called BeautifulSoup [1], developed in Python. We organized the data into a dictionary with topic_id as the key and the set of documents in the topic as its value, organized by document_id. We have also retained parameters like date time, title and narrative, which may provide a good basis for our summarization system.

4.2 Content Selection

The two main components in our selection module includes the preprocessing stage, the topic modelling component using LDA, sentence scoring using a combination of LDA, TF-IDF and concrete-

[1]https://www.crummy.com/software/BeautifulSoup/

ness scores (Brysbaert et al., 2014), and selecting sentences so that as much important content as possible is represented in the final summary.

4.2.1 Preprocessing

The preprocessing stage was motivated by the main content selection method - topic modeling with Latent Dirichlet Allocation (LDA) (Blei et al., 2003), which benefits from as little noise as possible in the input data. We used SpaCy[2] processing pipeline to obtain various properties of the sentences.

Our preprocessing for topic modeling simply consists of the following steps:

- stopwords removal;
- lowercasing and lemmatization;
- sentence segmentation and tokenization;
- removing punctuation;

This ensures that important topical concepts are not obscured by morphology and increase the performance of LDA.

To retain as much information as possible to further used in downstream tasks, we summarized the properties and various annotations and statistics of the input sentences and grouped them by the *topic (document set) identifier*. We filtered out *interrogative* sentences as those that would be less likely to represent the main content of the news articles. The annotations and statistics include *named entity recognition, part-of-speech tagging, dependency parsing, noun chunks, sentence lengths, sentence index, document index,* and *total number of sentences* in corresponding documents. The indices and lengths are useful in information ordering and content realization, while POS-tagging, dependency parse and named entity recognition are assumed to serve as heuristics and help take advantage of the properties of news genre and weight the candidate sentences accordingly. Figure 3 gives the overview of this component.

4.2.2 Topic Modeling

Latent Dirichlet Allocation (LDA) is a generative statistical modeling technique that allows us to estimate unobserved (latent) topics from an observed distribution of documents (bag of words). We have used the *gensim* [3] package which provides both single-core and multi-core (parallelized) implementation of the LDA model. We have used the default value of $\{$1/number of topics$\}$ for the hyper-parameter *alpha*.

For each set of documents, LDA scores each sentence according to 3 (an arbitrary chosen number) topics. Each sentence is then assigned the topic identifier (topic_id - an integer 0, 1 or 2) that received the highest score, and the overall score across three topics. These scores are used for the computation of a total sentence score based on a combination of LDA, TF-IDF and concreteness.

4.2.3 Parameter tuning

One of the drawbacks of LDA is that the latent topics are not always comprehensible in a semantically meaningful or ordinal way, these topics are usually only inferred after manual evaluation. We also observed that the document-topic distribution is not always consistent and is usually skewed towards one of the topics. This makes it restrictive towards extracting sentences from the corpus in accordance with the document-topic probability distribution without under-representation from other topics. We also observed a high overlap of relevant terms (we selected top 100 terms) between topics, these terms/words usually represent the functional words or constant theme across the corpus. To counter these effects, we use the LDA computed topic-term probability (*phi*) as a measure of importance. This approach emphasizes the terms that have a high overlap among the documents in one document set and thus efficiently models the topical structure of a document set.

4.2.4 Sentence scoring

LDA gave us reliable scores to select the most content rich sentences based on the latent topics distribution over sets of related documents. We considered other metrics to score the sentences that are known to be performant for extractive summarization such as TF-IDF and concreteness scores. Both methods were implemented but kept for the future iterations of our system. TF-IDF (term frequency-inverse document frequency), which reflects the importance of a lexical unit to a document in comparison to other documents. This score was computed using *TfidfVectorizer* from *sklearn* module.

[2]https://spacy.io/

[3]https://radimrehurek.com/gensim/models/ldamodel.html

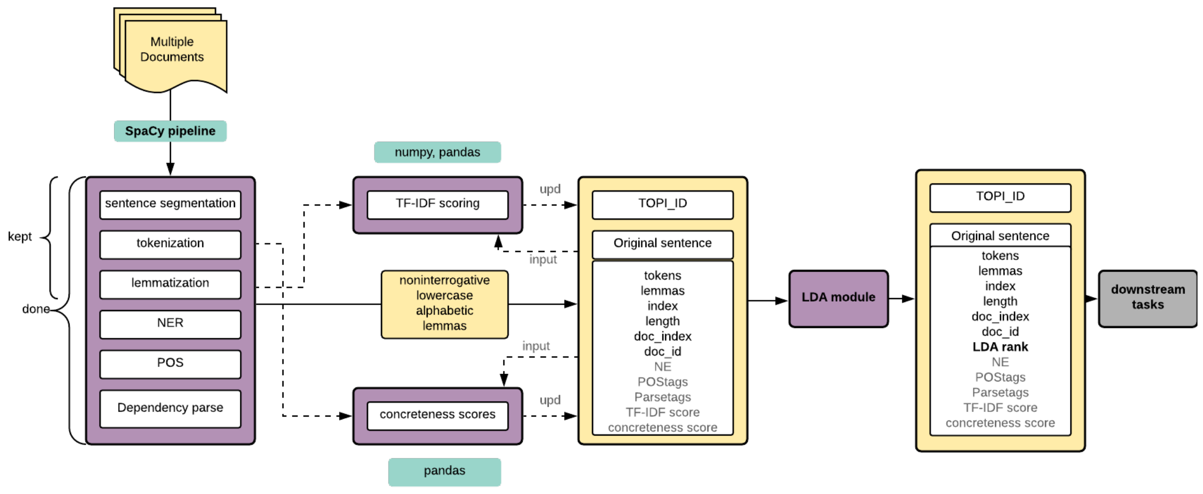

Figure 3: The preprocessing module

Concreteness scores are obtained from the dataset created by (Brysbaert et al., 2014). This dataset contains the concreteness ratings for 40 thousand English lemma words collected in a behavioral experiment with Amazon Mechanical Turk. The scores were summed for each sentence. This metric is be a good proxy for selecting the candidate sentences specifically from the news articles since we expect the relevant information to be written in accessible and concrete language. Concreteness also penalizes first-person statements, which reduces the number of sentences containing direct speech selected as relevant (but not completely excludes them since they can contain highly relevant terms).

The three metrics were then used in combination to compute a total sentence score, which is a product of TF-IDF, LDA and concreteness score normalized by sentence length. All three scores take values between 0 and 1 (TF-IDF and concreteness are rescaled, LDA is already in this scale) and thus they are equally weighted. The three metrics - LDA scores that rank the underlying content, TF-IDF scores that rate the representative power of vocabulary items, and concreteness scoring that works in line with the genre - brought notable improvements to our sentence scoring algorithm.

4.2.5 Sentence selection

To ensure that the selected sentences represent as much content as possible, we balanced the sentence selection across three topics. For each topic per document set, an equal number of sentences were selected based on their combined scores. This helped to reduce redundancy among the selected sentences and avoid the domination of one topic. The system in our final configuration produces 18 candidate sentences for each document set. Overall, the combination of various scores and a balanced topic selection brought significant improvements in ROUGE scores.

4.3 Information Ordering

Our information ordering module takes as input a list of 20 sentences selected for by the content selection module. First, we remove redundant sentences by setting a *redundancy threshold*. Second, we find the most coherent summary by permuting the sentences and computing a *coherence score*. We pick the ordering that gives us the highest coherence score. Both these measures rely on cosine similarity, and thus require a vector representation of the sentence.

4.3.1 Sentence Embeddings

We use 768-dim BERT embeddings, extracted through Hugging Face from a model that has been fine-tuned on a paraphrase detection task `bert-base-cased-finetuned-mrpc`[4] (Devlin et al., 2018).

4.3.2 Redundancy Threshold

To select a redundancy threshold, we annotate a small subset of training sentence pairs with the

[4]https://huggingface.co/bert-base-cased-finetuned-mrpc

labels *redundant* and *coherent*. We then embed our sentences using one of the methods mentioned previously, and calculate pairwise similarity across pairs of sentences. We then compute the average cosine similarity score within each set. As a starting point, we select a threshold that maximally separates the two sets. Previously, we refined the threshold by observing changes in ROUGE-2 scores. For the final deliverable, the threshold was refined by manually examining the outputs to ensure that the threshold filtered out redundant material. After removing redundant sentences using this threshold, we calculate a coherence score among different orderings of the remaining sentences.

4.3.3 Coherence Score

To find the most coherent summary among the remaining non-redundant sentences, we permute the sentences and calculate a coherence score for the order (Foltz et al., 1998). The coherence score is the averaged pairwise cosine similarity, as shown in (Barzilay and Lapata, 2008):

$$\text{coherence}(T) = \frac{\sum_{i=1}^{n-1} \cos(S_i, S_{i+1})}{n-1}$$

The information module returns a list of sentences containing the ordering with the highest coherence score.

4.4 Content Realization

For our content realization module, we use heuristics to remove unnecessary information and to patch any punctuation errors. The following heuristics from previous deliverables stayed in our system:

1. Ignore sentences with fewer than 8 words
2. Ignore sentences with greater than 50 words
3. Remove parenthetical expressions (material enclosed between parentheses, brackets, and hyphens)
4. Remove unnecessary phrases (“however”, “also”, “at this point”, “as a matter of fact”)
5. Remove ages (“aged 24” and “..., 24, ...”)
6. Remove attributive clauses containing “said”, ‘say”, ‘report”, “state”, “according”

Heuristic (2) was informed by experimentation with various upper word limits. See Table 1 for the ROUGE-2 scores obtained from setting different upper word limits for sentences. Heuristics (3)-(6) were modelled after those in CLASSY 2006 (Conroy et al., 2006).

Previously, heuristics were included on the basis of whether they increased ROUGE-2 Average Recall. For this deliverable, we focused on manually assessing output summaries, and we added or refined heuristics while focusing on the goal of improving readability. To that end, following heuristics were updated or added:

1. Remove adverbs, except sentence-final adverbs and those found in a *list of exceptions*
2. Remove sentences with any quotes
3. Remove sentences with capslocked words
4. Remove days of the week and replace with determiners (Tuesday’s incident $\rightarrow$ the incident, Tuesday night $\rightarrow$ one night)
5. Remove “, which. . . ” clauses at the end of the sentence
6. Remove “but” at the beginning of a sentence

Heuristic (1) was informed by our manual assessment of the summaries, which revealed that the strategy of removing all adverbs except sentence final adverbs is too general, and leads to a number of readability issues. For example, negation tokens like *not* and *n’t* were being stripped. We made our removal strategy more targeted by creating an *exception list* based on the reviewed summaries. If the adverb was found in this list, then it was not removed.

Our algorithm adds sentences to the summary in the order of their total sentence score, with the highest scored sentence first. But, should the addition of a sentence put us over the 100 word limit, we opt for the next best sentence. This is done to promote readability, since we do not want to have sentences truncated at 100 words.

5 Results

Our final system has following (arbitrarily chosen) hyperparameter settings:

Upper word limit	ROUGE-2 (Average Recall)
60	0.04584
50	0.04741
40	0.0404
30	0.01477

Table 1: Effects of changing the upper word limit for sentences on ROUGE-2 Average Recall.

- **LDA topics**: 3
- **Number of selected sentences**: 20
- **Similarity threshold**: 0.94

We use a standard implementation of ROUGE to evaluate our summaries. The metric measures overlap between automatic and human summaries. Our primary metric is ROUGE-2 Recall, which measures bigram overlap. ROUGE-1 Recall, on the other hand, measures unigram overlap. A higher ROUGE score signals better overlap with the human summary. This metric has been shown to correlate well with human assessments of responsiveness and to work well for extractive summaries (Rankel et al., 2011).

A comparison of our system to baselines and to previous submissions is shown in Table 2. The LEAD baseline takes the first 100 words as the summary, while MEAD is an exemplar centroid-based summarization system (Radev et al., 2001).

On the *devtest* data, our D4 system shows a significant improvement over our D3 system. Our D4 system also outperforms the LEAD baseline, and marginally outperforms the MEAD baseline.

	ROUGE-2	ROUGE-1
Devtest		
LEAD Baseline	0.05376	
MEAD Baseline	0.05927	
D2 System	0.04340	0.18334
D3 System	0.05427	0.22305
D4 System	0.06179	0.22500
Evaltest		
D4 System	0.06676	0.23833

Table 2: Comparison of our D4 system to baselines and previous submissions. ROUGE-2 Average Recall and ROUGE-1 Average Recall are reported.

6 Discussion

6.1 Effects of Hyperparameters

Table 3 shows the effects on *devtest* when the values of two hyperparameters are varied. The two hyperparameters are the number of selected sentences and the similarity threshold. The number of selected sentences varied by steps of 10 and ranged from 10 to 50, while the similarity threshold varied by steps of 0.2 and ranged from 0.91 to 0.99.

We find that when we have the fewest number of sentences, the highest similarity threshold works best. The lower the similarity threshold, the more likely it is that a greater number of sentences will be filtered out. We hypothesize that when the set of sentences is small to begin with, a high similarity threshold prevents us from removing too many sentences, which in turn prevents against summaries that are far below the word limit.

Once we reach the mid-point of 30 sentences, however, the mid-point similarity threshold of 0.95 performs the best. With this configuration we achieve our best overall ROUGE-2 Average Recall, **0.06805**. We also appear to reach a plateau at this point. Even as we increase the number of selected sentences increases beyond 30, the 0.95 threshold consistently yields the the best score of 0.06805.

Interestingly, the threshold of 0.97 gives the best average performance across the different number of selected sentences.

6.2 Ordering and Selection

The qualitative analysis of the summaries revealed several problematic areas. First, some summaries contain sentences that are either irrelevant or lacking coherence with the surrounding context. One reason for this is the limitations of LDA, which favors longer and more topic-rich sentences and has no internal mechanism to differentiate between the sentences with the same scores. As a result, content realization may select the sentences of a suitable length but leave out the most relevant sentences that are longer. These issues can be tackled by scoring the LDA subtopics, distributing the sentences across the subtopics and controlling that the selected sentences cover the most highly ranked distinct subtopics.

	Similarity threshold					
# Selected	**0.91**	**0.93**	**0.95**	**0.97**	**0.99**	*Mean*
10	0.03979	0.046790	0.057430	0.06375	**0.064930**	0.054538
20	0.03921	0.048710	0.066630	**0.06703**	0.065720	0.057460
30	0.04188	0.051840	**0.068050**	0.06699	0.066360	0.059024
40	0.04389	0.055300	**0.068550**	0.06751	0.066460	0.060342
50	0.04488	0.059690	**0.068050**	0.06747	0.066010	0.061220
Mean	0.04193	0.052466	0.065742	**0.06655**	0.065896	0.058517

Table 3: ROUGE-2 Average Recall on *devtest* when we vary the similarity threshold and the number of selected sentences. The highest score in each row is bolded. The highest similarity threshold works best when we have the fewest number of sentences. For 30 sentences and over, the slightly lower threshold of 0.95 performs best.

6.3 Redundant Fragments

Another problem is redundancy of sentence fragments. Being the main focus of this release, sentence-level redundancy was almost completely eliminated (only 6 summaries out of 46 still contain redundant sentences), but cosine similarity is much less efficient with sentence fragments that do not significantly change the total score being negligible proportionally to the sentence length. Although redundant fragments were spotted in less than 22% of the summaries, we expect that using dependency parses or phrase-structure level instead of punctuation as a heuristic would help solve this problem.

6.4 Redundant Attribution, Coreference, Connectors

The issues with redundant attribution, unresolved coreference and sentence connectors were not yet given sufficient attention and expectantly propagated to our summaries. Finally, the interplay between different modules remains a problem to solve. For instance, the content realization is currently centered around redundancy reduction, and thus the improvements to content selection resulting in less repetitive made the former inefficient and harmed the performance. We are yet to discover the best configuration of the different modules in the pipeline with the help of ablation study of different components and hyperparameters. Overall, about 46% of the summaries are of sufficient quality and need only minor improvements, which is a promising result.

6.5 Error Analysis

In our error analysis, we look at some specific output summary examples from the devtest and evaltest data that highlight the general categories of errors we observed in the results of our D4 submission.

6.5.1 Sentence Length and Cohesiveness

We noticed that some of our summaries included short sentences that were out-of-context relative to the rest of the summary. An example, below, shows a summary in which the second sentence is not cohesive with the first. We suspect this is due to the large number of long sentences that appear in the original documents and are favored by LDA. If our algorithm does not find an additional sentence that can be added to the summary, while staying under the word count, it will add the shortest selected sentence that will fit.

> **Devtest-D1003**
> Three giant pandas living in Beijing Zoo will be sent to Wolong of Sichuan Province, southwest China, and three others in Sichuan will fly to Beijing, under an exchange program aimed to maintain the biodiversity of the giant panda population.
> China to complete country's first blood bank for pandas.

6.5.2 Summary Length

In our manual evaluation of our summaries, we used summary length as one of the metrics. Our aim was to get our summaries as close to 100 words as possible without going over the word limit. Our solution does not fully address the problem of short summaries, so this would be an area

Data	Average Summary Length
D2	76.283
D3	84.5
D4-devtest	73.848
D4-evaltest	72.341

Table 4: Average number of words in summaries by data set

to keep exploring in further work. Table 4 shows the average summary length for each deliverable. Our D3 submission achieved the highest average summary length, with shorter overall summaries in D4 for both data sets.

The following examples show one-sentence summaries from our D4 submission outputs:

Evaltest-D111
An 18-year-old student opened fire in a high school in southern Finland, killing eight people including the principal before turning the gun on himself.

Evaltest-D1125
If the polar bear were listed as a threatened species, all federal agencies would have to ensure that anything they authorize that might affect polar bears will not jeopardize their survival or the sea ice where they live.

Though these single sentences provide key pieces of information for the summary, they are well below the 100-word limit, which we believe negatively impacts our ROUGE scores.

6.5.3 Issues with Heuristics

While we tried to make our sentence compression heuristics as generalized as possible, there were some cases where our heuristics went too far and eliminated essential sentence components. One example of this was in our handling of adverbs. We originally eliminated all adverbs, but had to go back and add specific cases after we noticed examples of adverbs being removed in a way that detracted from the sentence structure and led to poor readbility. We discovered that we did not address all these cases, as is shown in the example below. The adverb "ago" is removed where it should not be.

Devtest-D1005
Three years ~~ago~~, Bush limited federal funding of embryonic stem cell research to the 78 stem cell lines in existence.
A new robotic microscope that follows a brain cell in the lab from a normal state to its death has led researchers to a surprising finding that appears to debunk a long-standing theory about Huntington's disease.

6.5.4 Coreference

Our D4 submission has remaining issues with coreference that detract from the readability of our summaries. In some cases, there is repetition of referent expressions, as in the example below. Australia is repeated four times, and this would be better handled with coreference.

Devtest-D1022
Humane Society International was denied permission by Australia's Federal Court last week to sue the Japanese whaling company for killing hundreds of whales in Antarctic waters the Australian government has declared a whale sanctuary.
Australia vows unprecedented drive to curb Japan whale kill.
Australia last week warned Tokyo it was embarking on an unprecedented diplomatic campaign to dissuade Japan from trying to escalate its killing of whales.

On the other hand, we also had some summaries that included pronouns without first naming the full title of their coreferential named entities. These cases also lead to poor readability.

Evaltest-D1022
They pointed to hackers' powerful attacks this week on the computers that manage global Internet traffic as evidence of the mounting threat from online criminals.
The effort would not improve response time, the company said, but would make it possible to recognize and contain Internet attacks more quickly.

Evaltest-D1140
At least eight people died in floodwater or flood-related incidents in three northern districts, and about 1 million people were marooned by floodwater in

> about ten districts in northern, northeastern and central parts of the country. The government's flood forecasting and warning centre said the situation could worsen in the next few days as major monsoon and Himalayan-glacier fed rivers are expected to crest.

Overall, we saw improvements in readability with our manual evaluations in each deliverable, but there is still room for improvement, in particular for the categories discussed in this section.

6.6 Other Algorithms

We have experimented with graph-based methods for content selection - specifically graph-based TextRank (Barrios et al., 2016) algorithm and sentence vector encoder skip thought vectors (Kiros et al., 2015). TextRank ranks the nodes (sentences) in a graph to determine the top ranked sentences. However, if the algorithm does not converge at a node it raises an Exception, and hence renders the whole document set without any top ranked sentence. For this reason we did not incorporate it into our final model. We also tried using skip-thought vectors in our content selection but due to scarcity of compatible implementations we had to discard the algorithm.

7 Conclusion

An extractive multi-document summarization system was successfully implemented using topic modeling with LDA as the main content selection method and surface level filtering of the summaries produced by the system. The end-to-end system achieved the ROUGE-1 and ROUGE-2 scores comparable with the baselines and outperformed the baseline in most cases. The scores show that our approach has the potential to yield good results. With productive tweaks to our systems, such as increasing the top sentences from the LDA algorithm and using concreteness scores, we have been able to increase the readability of the summaries. While there still persist some unresolved issues, we find that our summaries are highly readable and discernable. As we improve our system, we see a consistent improvement in our scores and overall summary readability, which gives us the confidence in our implementation.

References

Federico Barrios, Federico López, Luis Argerich, and Rosa Wachenchauzer. 2016. Variations of the similarity function of textrank for automated summarization. CoRR, abs/1602.03606.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. Computational Linguistics, 34(1):1–34.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. Journal of machine Learning research, 3(Jan):993–1022.

M. Brysbaert, A.B. Warriner, and V. Kuperman. 2014. Concreteness ratings for 40 thousand generally known english word lemmas. Behavior Research Methods, 46(1):904–911, August.

John M. Conroy, Judith D. Schlesinger, Dianne P. O'Leary, and Jade Goldstein. 2006. Back to basics: Classy 2006.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Peter W. Foltz, Walter Kintsch, and Thomas K Landauer. 1998. The measurement of textual coherence with latent semantic analysis. Discourse Processes, 25(2-3):285–307.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. CoRR, abs/1506.06726.

Dragomir R Radev, Sasha Blair-Goldensohn, and Zhu Zhang. 2001. Experiments in single and multidocument summarization using mead. In First document understanding conference, page 1À8. Citeseer.

Peter A Rankel, John Conroy, Eric Slud, and Dianne P O'leary. 2011. Ranking human and machine summarization systems. In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 467–473.