Automatic Summarization

Matthew Calderwood
University of Washington
calderma@uw.edu

Kirk LaBuda
University of Washington
kwlabuda@uw.edu

Nick Monaco
University of Washington
nickmon@uw.edu

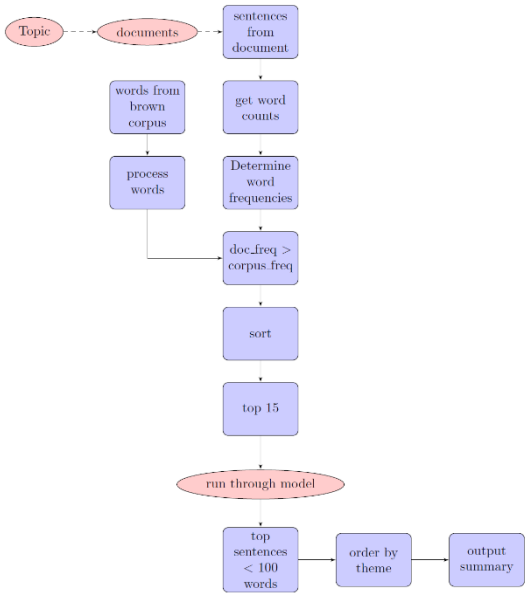
Abstract

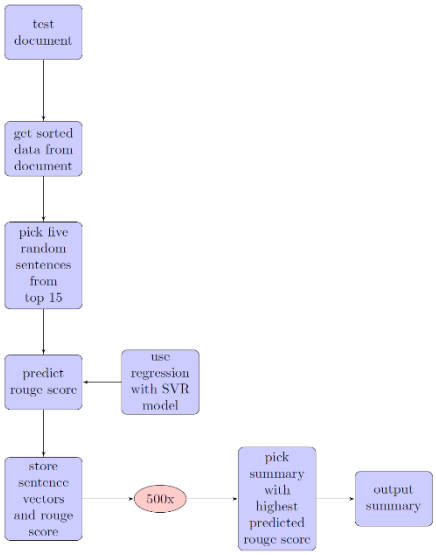
The purpose of this project is to participate in a shared task of automatic summarization. Our goal is to break a large problem into manageable subtasks, incrementally improve our system, and evaluate the results at each step. The data comes from the TAC 2010 Guided Summarization shared task, and ROUGE is used for evaluation. Our system has three main components: preprocessing, content selection, and content realization. Our model is extractive, so focus falls primarily on content selection. Preprocessing is used to collect the text and remove any irrelevant information. Content realization uses a tree based approach that attempts to prune unnecessary nodes, thus allowing more information in the summary.

1 Introduction

With the vast amount of publically available text information on the internet, one of the most important uses of language processing is to help us query and extract meaning from these large repositories (Jurafsky and Martin, 2008). This project aims to implement a working system for automatic summarization, based on the information presented in class and our knowledge from the program thus far. According to Mani (2001), the goal of automatic summarization is to take an information source, extract content from it, and present the most important content to the user in a condensed form and in a manner sensitive to the user's or application's needs. While our implementation will not have any practical applications outside of the classroom, we hope to improve our system over time, to the point where it can produce coherent summaries.

2 System Overview





Our system architecture is fairly similar to the diagram in the course notes. First, topics and their relevant document IDs are extracted from the TAC 2010 Guided Summarization task data file.

News articles for each topic are located by searching for the document IDs within the aggregated data sets of different publishers. Once found, the article is parsed to extract only the plain text. During preprocessing, metadata and other irrelevant info are discarded.

Each article is segmented into sentences and words. The sentences are kept intact since our model is extractive, while the word segmentation is used to store frequency counts for the words within each document.

We decided to make use of NLTK's stopword set and the Brown Corpus in Python. We used the WordNet lemmatizer to lemmatize the Brown corpus and access the frequency distribution based on these lemmas.

From here, we computed and stored the log probability of every sentence in our topic set from word frequencies within the topic documents. We compared our computed likelihood of every sentence with the probability based on the lemmatized Brown corpus probability using tf*idf. This gave us a relative idea of how important or thematic a given word was to our summary.

Where the difference was greatest, we knew that we had a sentence that was valuable for our summary. We then decreased the importance of these words to try and eliminate repeats of the same information. We stored this computed difference for all sentences. Then, we use regression with an SVR model to choose the highest scoring summary out of the top 15 sentences.

Before the 15 sentences are scored, they are ran through content realization, which uses a machine learning approach to remove nodes from each sentence's parse tree. This step is done before scoring in case any important words are removed.

After finishing our summary, the first sentence is chosen using a theme-based approach. After the first sentence is assigned, the remaining sentences are ordered by their similarity to the first one.

After this reordering, we simply output our summaries and used our rouge script to evaluate the results.

3 Approach

Our system is split into distinct components, each one handling a different sub-problem of the summarization task. The components are run linearly, with the output of one providing the input for the one that follows.

3.1 Preprocessing

The process begins by extracting the news article text from the TAC 2010 data. Document IDs and article headlines are also extracted. To facilitate the tf*idf computation, any irrelevant information is removed. This includes metadata, datelines, authors, and sentences less than four words. Following this, the article text is segmented into sentences. Furthermore, all the words are segmented and stripped of punctuation to provide word counts.

3.2 Content Selection

As stated above, we selected content based on our tf*idf computation. Our system computed term frequency in the document set for a given topic, and computed the probability of a given sentence. Separately, we computed the probability of that sentence based on a lemmatized version of the Brown corpus and NLTK's frequency distribution tool. We then save the 15 highest scoring sentences and combine them into combinations of five sentence summaries. Content realization is done once the 15 sentences are selected rather than at the end, which will be explained below. We then randomly select 500 among the possible combinations and run those through our SVR model to determine the predicted rouge-2 score. Each summary becomes a data point in 10 dimensional space (x0...x4 = tf*idf score of candidate sentences, y0...y4 = position of sentence in article for each sentence). We then ran this against our SVR model to find the predicted ROUGE-2 score. At the end of the process, the summary/data point with the highest predicted ROUGE score is selected as the summary for this document set.

We investigated a few ways to improve our ROUGE scores. One of our ideas was to add more features other than tf*idf score and sentence length. However, we could not find any features that increased our overall scores. Therefore, we decided our 10-dimensional vector would be sufficient. Another idea was to normalize the sentence position values. That is, we took note of the fact that a sentence occurring in position 2 out of 4 sentences (value = 2/4) would probably have a different relationship to the first sentence than the second sentence in a 40 sentence article (value = 2/40). This change ultimately improved our ROUGE results.

We approached model creation and selection in the following way. First, we divided our data into 37 training samples and 9 testing samples, with each sample being one topic. For the training data,

we used the above methodology (selecting the sentences and creating the summaries in the same manner to create the 10 dimensional vector) then used the ROUGE script to run over the created summary to find its actual ROUGE score. We then used this as a training example. The format is:

<actual rouge score> <tf*idf sentence1>:value <position sentence1>:value...

We then trained on a scaled version of these data points using libsvm to create a nu-SVR model. After testing various kernels, we concluded that the RBF kernel produced the best results for ROUGE score prediction. For the test dataset, we use the model obtained through training to predict the ROUGE-2 score for each generated summary then pick the summary with the highest projected score.

3.3 Information Ordering

For information ordering, we use a theme-based approach to select the first sentence. We reasoned that a good theme would be a frequently occurring term within the document set, and that a good theme would logically be a noun. We used NLTK's POS tagger to ensure that the thematic word for an article would be either NN, NNS, NNP, or NNPS.

Once the sentences have been selected for each summary, the last step before output is information ordering. We tokenized and kept token counts for both individual summaries and the document set from which they came. We reasoned that the best theme for a sentence would be the word that occurred most frequently in that sentence. We thus assigned a theme to every individual sentence in our summary. We then checked the word count for that theme within the document set, and assigned the sentence with the highest theme score as the first sentence.

After the first sentence is chosen, the theme score for other sentences is discarded. Rather, they are ordered according to their similarity to the first sentence. Every sentence was converted into a word-count vector, and its cosine distance from the first sentence was measured using the SciPy package for Python. Since this is a distance measure, sentences are arranged from least distant to most distant. This approach gave an appreciable improvement to readability.

3.4 Content Realization

We decided to take a machine learning based approach for content realization. This is made possible by a sentence compression corpus, which contains pairs of an original sentence and the same sentence with unimportant information removed (Clarke & Lapata, 2008). We also decided to use tree based compression rather than word based, since it seems more intuitive to remove constituents (tree nodes) rather than individual words.

For training, each sentence pair was extracted from the corpus and parsed using Stanford CoreNLP. The resulting trees were stored and read using NLTK. By comparing the tree structures, each node was labeled as keep, omit, or partial. A variety of features were used to provide a context for each node. These include the node's POS, its (grand)parent's POS, its siblings' POS, and the words in its leaves. There were also features that checked if the node was the left-most child of the parent, or if the node contains negation. The training was done using MALLET, and the trainer used was MaxEnt.

The content realization step was done during content selection, specifically after the 15 sentences are selected for each summary. This was done so that the sentences would be scored after they were compressed. If compression occurred later, it might remove words that gave a sentence a high score.

Once the 15 sentences are selected, they are parsed and each node is labeled based on the model obtained from training. Due to time constraints, a conservative approach was taken where all the children of a kept node are kept, regardless of their label. Ideally, different combinations of node removals would be tested and ranked using a language model. This way, grammaticality could be retained without sacrificing compression.

4 Results

Our ROUGE scores drop consistently from ROUGE1-ROUGE4. ROUGE1 scores are our highest, and our Average_P value was consistently the highest out of F, R and P. Our average R-scores are shown below, which compares past results with our current results:

	D3	Devtest	Evaltest
ROUGE-1	0.12951	0.15066	0.16779

ROUGE-2	0.02431	0.02844	0.02968
ROUGE-3	0.00612	0.00717	0.00705
ROUGE-4	0.00198	0.00265	0.00212

5 Discussion

We were glad to see improvement to our ROUGE scores with each iteration of our project. We were able to gradually expand and improve each component of our system. Despite our success, there are some areas we would have liked to improve given more time.

In particular, content realization was the last major feature implemented, and it was somewhat experimental in nature. While it was helpful overall, it often made sentences less readable. As mentioned earlier, using a language model would help with this problem. However, there are other ways it could be improved. There are some cases where a node should always be removed or kept, so adding rule based overrides would catch these instances. In fact, a purely rule based approach may fare better. Other summarization systems have found success in this approach. Furthermore, training on the compression corpus makes it difficult to predict whether nodes with the same parse tree context are kept or removed (only words provide a clue, which may be too specific).

5.1 Error Analysis

Despite our gradual increase in ROUGE scores, there were two minor exceptions: ROUGE-3 and ROUGE-4 recall from the evalset. This decrease is understandable, since our model aimed at optimizing ROUGE-2 scores, with the assumption that improving ROUGE-2 would quite probably bring up all the other ROUGE scores as well. This is not a tenuous assumption, but it is also not foolproof. In addition, the evaluation set contained new data, which always has the potential to show shortcomings in the system that didn't occur in the training and development stages.

Additionally, adding content realization to our system slightly decreased some of our scores. Presumably, it was removing important words or phrases from several sentences or removing large chunks altogether. Adding further checks on removal or switching to a rule based system may prevent these words from being removed.

6 Conclusion

This project was an attempt to implement an automatic summarizer as part of a shared task. Additionally, it was our goal to work collaboratively to incrementally improve our system. Our scores went up at each step, so we are satisfied with our results. If given more time, we feel confident we could have continued this trend.

By participating in this shared task, we have learned which aspects of our system worked well, and also the aspects other teams did well. This provides insight into which approaches achieve good results, and the variety of approaches one can take to solve a problem. Shared tasks such as this one may be instrumental for improving the state of automatic summarization and NLP problems in general.

References

Chang, Chih-Chung and Lin, Chih-Jen. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

Clarke, J and Lapata, M. *Global Inference for Sentence Compression: An Integer Linear Programming Approach*, Volume 31, pages 399-429, 2008.

Jurafsky, Daniel and Martin, James H. 2008. *Speech and Language Processing*. Prentice Hall, Upper Saddle River, NJ.

Li, Chen, Yang Liu, Fei Liu, Lin Zhao, and Fuliang Weng. *Improving Multi-documents Summarization by Sentence Compression Based on Expanded Constituent Parse Trees*. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014.

Mani, Inderjeet. 2001. *Automatic Summarization*. John Benjamins Publishing, Philadelphia, PA.

Smola, Alex J. and Schölkopf, Bernhard Schölkopf. A Tutorial on Support Vector Regression*. *Statistics and Computing 14*: 199–222, 2004. http://alex.smola.org/papers/2003/SmoSch03b.pdf

Yu, Pao-Shan, Chen, Shien-Tsung and Chang, I-Fan. Support Vector Regression for Real-time Flood Stage Forecasting. *Journal of Hydrology*, 328 (3–4), p. 704–716, 2005.