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

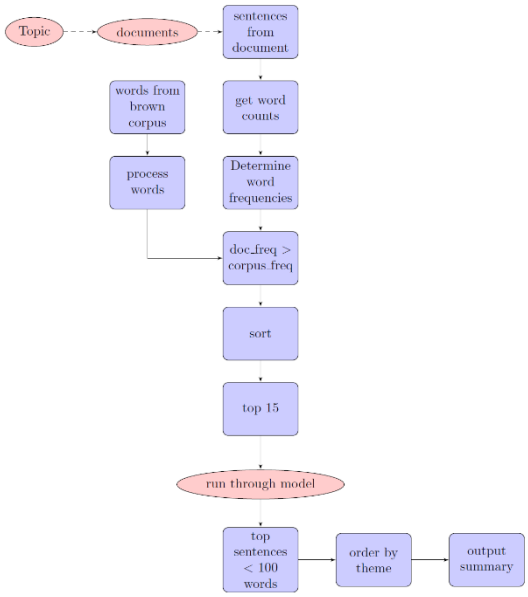
Abstract

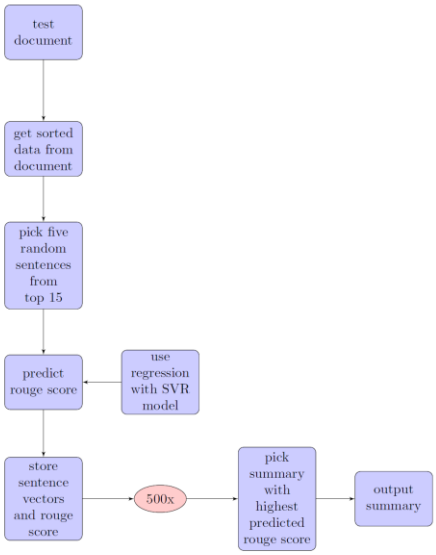
The purpose of this project is to participate in a shared task of automatic summarization. Our goal is to break a large problem into manageable subtasks, incrementally improve our system, and evaluate the results at each step. The data comes from the TAC 2010 Guided Summarization shared task, and ROUGE 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 weed out any irrelevant information. Content realization is trivial, since we simply output the extracted sentences in chronological order. However, we may modify this process in order to shorten sentences or improve the readability of our summaries.

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. Paragraph markers are discarded, though this information may be used in the future.

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 that it would be convenient to avail ourselves 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.

After finishing our summary, the sentences are ordered using a theme-based approach. A theme is assigned to each sentence, and the sentences are arranged from most prominent theme to least prominent theme.

After this reordering, we simply output our results 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, the article text is stripped of all tags and 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. 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 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. We used a linear nu-SVR for D3 because the other kernel types take several hours longer to run (polynomial with standard settings takes about half a day) and we wanted to be able to change settings on the base system more easily and see the results. Training for other kernel types is underway for D4. 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 this deliverable, we decided to use a theme-based approach for our information ordering. 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 realization and 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 ordered sentences in decreasing order by their theme's score within the document set.As discussed above, after the content selection phase we ordered the sentences within the summary based on the date of the article they occurred in. This assignment didn't have too much required for content realization - we simply made sure that our summaries were less than 100 words as requested.

3.4 Content Realization

Currently, our system does not make any modifications to the summaries once the sentences have been selected and ordered. This is the next area we will address, since many of our summaries contain long sentences, which doesn't work well when the summaries are limited to 100 words. Therefore, using sentence compression techniques will be beneficial in this regard.

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:

	D2	D3
ROUGE-1 Average_R	0.10987	**0.12951**
ROUGE-2 Average_R	0.01891	**0.02431**
ROUGE-3 Average_R	0.00502	**0.00612**
ROUGE-4 Average_R	0.00129	**0.00198**

5 Discussion

Class 7 has some baseline ROUGE-2 scores for LEAD and MEAD. It seems the average of the LEAD/MEAD scores is ≈.05. Our average ROUGE-2 result was ≈.02 (.01891). Our results have improved since our last evaluation, but we would like to continue improving these scores through further experimentation. For content selection, we will explore using other kernels with SVR to see which yields the best results. Feature selection could also use some more tweaking and improving. In terms of content realization, implementing some form of sentence compression is the next logical step. Lastly, our system's runtime could use some optimization. With the new machine learning implementation and the 500 iterations per summary, the whole process takes about 30 minutes to run. This makes it challenging to test various tweaks to our system.

6 Conclusion

This project is a work in progress. At present, our system architecture is more fleshed out from the last evaluation, and we have managed to improve the results. The challenge now is to refine our system by finding areas that could be improved or added to. Our goal is to work collaboratively to implement NLP ideas and techniques, improve our automatic summarizer, and learn from our experience in a shared task.

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

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

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.