Summarization Project Report: Deliverable 2

Long Cheng, Sheng Bi, Catherine Wang, Vicky Xiang, Carrie Yuan
University of Washington
`{lcheng97, shengbi, qywang, xinyix7, jiayiy9}@uw.edu`

Abstract

1 Introduction

2 System Overview

The summarization system is designed to condense text content into concise summaries through a multi-stage process. The system architecture consists of three main components: Content Selection, Information Ordering, and Content Realization.

Content Selection

- The first stage of the system is Content Selection, where various algorithms are applied to identify the most important information from the input text.
- Three algorithms, namely LLR, BasicSum, and LexRank, are employed for Content Selection. These algorithms analyze the text and prioritize sentences or phrases based on their relevance, importance, and semantic connections.
- The output of Content Selection is a set of selected sentences or phrases deemed most crucial for inclusion in the summary.
- Each algorithm's performance is evaluated using ROUGE scores, providing a quantitative measure of how well the selected content captures the essence of the original text.

Information Ordering

- The next stage of the system, Information Ordering, is responsible for structuring the selected content in a logical and coherent manner.
- This component arranges the selected sentences or phrases in a sequence that enhances the readability and comprehension of the summary.
- Information Ordering ensures that the summary flows smoothly and effectively communicates the key points of the original text.

Content Realization

- The final stage of the system, Content Realization, focuses on generating the actual summary based on the ordered content.
- This component may involve text generation techniques such as sentence fusion, paraphrasing, or abstraction to create a concise and informative summary.
- Content Realization aims to produce summaries that capture the essence of the original text while minimizing redundancy and irrelevant details.

Overall, the summarization system employs a modular architecture, allowing for the independent development and optimization of each component. The system's effectiveness is evaluated through both qualitative assessments of summary coherence and quantitative metrics such as ROUGE scores. As the system evolves, enhancements and refinements to each component will contribute to the generation of high-quality summaries.

3 Approach

3.1 Preprocessing

Our training, dev, and evaluation sets are taken from the TAC 2009/2010/2011 shared task data, which gives lists of documents organized by topic. Our data prepocessing involves two steps: first, we extract topics and documents from the TAC shared task data files through our script `extract.py`, then parse, format, and preprocess the data in each document file. This is implemented in the scripts `{training | dev | eval}_process.py`, which handle formatting and preprocessing for the training, dev, and evaluation splits respectively.

The extraction process involved parsing the TAC data files in XML format, then extracting topic and document IDs. Once `docIDs` are extracted, the local path for each document is generated based on the file name, year, and directory. This allowed us to get each file name for all documents in each split of our data.

Once we have the set of articles for our training, dev, and test data, we implement preprocessing by parsing the files from their original XML format and writing out plain text files for each document that contain the preprocessed data. Due to the variations in formatting of the original data, our preprocessing is split into three separate scripts for the training, dev, and evaluation data respectively. These scripts utilize `xml.etree.ElementTree` to parse the XML files, and manually modify the files (by adding `<DOCSTREAM>` tags) if they are not originally in standard XML format. For nonstandard file formats, such as some files in the dev split, we also had to reformat text including replacing `‘&’` with `’and’`, which may impact the summarization system. Once the XML data is parsed, each document is found by its `docID`, and relevant information such as headline and dateline are extracted. Finally, each document is segmented by sentence, and tokenized using `nltk.word_tokenize`. The output files are sorted into subdirectories by topic, and each outputted file is a plain text document that includes the headline, dateline, and segmented and tokenized text of the document body.

3.2 Content Selection

3.2.1 LLR (Log-Likelihood Ratio)

The LLR method implemented in this work is derived from the approach outlined by (Lin and Hovy, 2000). Initially, a background corpus is required. In our case, we utilized data spanning from September 21, 1997, to June 9, 2007, sourced from the directory TAC_2010_KBP_Source_Data/data/2009/nw/cna_eng. This background corpus was consolidated into a file named back_corpus_file.

In the LLR algorithm, our initial step involved tallying the total word count and the frequency of each word from the background corpus. Our approach entailed excluding punctuation marks, while the exclusion of stop-words was determined by the last argument of the command line parameters. During the evaluation phase, we will present ROUGE scores for both implementations, with and without stop-words. Given the nature of our multi-document summarization task, each document set, comprising 10 files, was treated as an independent input. We iterated through each document set within the input data directory to generate a summary for each document set (topic).

We followed a similar counting procedure for each input as we did with the background corpus. Subsequently, we applied the primary LLR technique to identify "important words" whose ratio surpassed the confidence level (set at 0.05 in our approach). Next, we prioritized sentences based on the quantity of "important words" they contained. Ultimately, we selected a subset of sentences with the highest weights to compose the summary. It’s worth noting that in our approach, the length of the final summary was restricted to 100 words or fewer.

In summary, the LLR method contains the following steps:

- Tokenization: Tokenize the text into words or phrases.
- Calculate Word Frequencies: Calculate the frequency of each word or phrase in the background corpus and each document set.
- Calculate LLR Score: Compute the LLR score for each word or phrase using the formula for LLR. The formula for LLR is typically based on the frequencies of the term in the document and in the corpus.
- Select Top Words: Select top words or phrases based on their LLR scores and the confidence level. These top words are more significant or relevant to the document.
- Select Top Sentences for Summarization: Rank the sentences based on the number of top words they contain. Choose a subset of sentences with the highest weights to form the summary.

3.2.2 sumBasic

The original paper (Nenkova and Vanderwende, 2005) investigates how the frequency information alone can effectively contribute to summarization. It starts by presenting a key finding that, when using word frequency as a content selection mechanism, the words that are used by almost all human summarizers tend to be high frequency ones and the words used in only few human summaries tend to be of low frequency (see table 2 of the paper). Also, a significant proportion of words used by only few human summarizers tend to have lower probabilities. These observations suggest the need for mechanisms that enhance the role of low-frequency words in sentence weighting, proposing the use of semantic content units (SCUs) for this purpose, that is, using atomic facts presented in a text.

The paper introduces sumBasic, an algorithm that studies the exclusive impact of frequency information on summarization, where SCU frequencies are utilized to determine sentence importance for inclusion in summaries. Apart from the use of the SCUs to calculate token probability and sentence weights, it included a feature where the probabilities of tokens in already selected sentences are reduced in order to discourage selecting similar sentences subsequently. This process repeats until the summary reaches a specified length, aiming to capture diverse and representative content from the source documents.

Our implementation inherits the algorithm suggested by the paper. However, as we do not have data of human-annotated SCU data as the original article does, we simply used word frequency to calculate token probabilities and sentence weights. To approximate SCUs' functionality, we're exploring improvements in preprocessing, currently experimenting with bigrams and tf-idf techniques.

In summary, our sumBasic algorithm consists of the following steps:

- Calculate Word Frequencies: Calculates the probability of each word in the document set, excluding stop words, based on frequency.

- Calculate Sentence Score: Identifies the sentence that has the highest average probability of its words.

- Select the top-scored sentence: Add the selected sentence to the summary.

- Update Probabilities: The probabilities of the words in the chosen sentence are updated (reduced) to avoid repetitive content in subsequent selections.

- Repeat until Done: Repeat the summary reaches the desired length, ensuring diverse and relevant content coverage.

3.3 Information Ordering

3.4 Content Realization

4 Results

4.1 Evaluation

For query-based tasks such as text summarization, researchers could evaluate how good a system by relying on two intuitive aspects: the distribution of divergence and similarity to the input (Louis and Nenkova, 2013).

To capture the similarity between the system summaries and human ones, we start with ROUGE to account for the number of summaries overlapping those in reference summaries. Using this metric, the overlapping ngrams is to be divided by the total number of ngrams in the reference summary and system summary, respectively for recall (R) and precision (P), and then produce a harmonic mean of P and R.

4.2 Evaluation Results

For each content selection system, we generate a directory of outputs that contain plain text summaries for each topic, for a total of 46 summaries. Using human-written model summaries as the gold standard, we calculate ROUGE scores for each system-generated summary. When multiple gold standard summaries are available, we take the average of the ROUGE score over each model summary for a given topic, giving us the most comprehensive evaluation. We then calculate the average ROUGE scores over each output directory, giving us the final evaluation scores for each system.

Since both unigram- and bigram- based algorithms are used in our content selection approaches, we choose to include ROUGE-1, ROUGE-2, and ROUGE-L scores when evaluating our system outputs. Table 1 presents the ROUGE recall values for each system.

A general pattern we observe is that the average recall values get closer to 0 (less similar) and

Content Selection Method	**ROUGE-1 Recall**	**ROUGE-2 Recall**	**ROUGE-L Recall**
SumBasic (unigram with stopwords)	0.2704799348	0.04039093478	0.1215869565
SumBasic (unigram without stopwords)	0.1333506957	0.00867423913	0.07359052174
SumBasic (bigram with stopwords)	0.1795537826	0.01769426087	0.1194731304
LLR (with stopwords)	0.1596591304	0.008082695652	0.0984761087
LLR (without stopwords)	0.1621166522	0.008096217391	0.09865563043
LexRank	0.1808198696	0.01334576087	0.1052659565

Table 1: ROUGE recall values for the outputs of each content selection method in our system compared to model summaries.

further from 1 (more similar) as we increase the n-gram size. We also notice that for the SumBasic method, stopword removal largely negatively impacted recall, but improved precision. This makes sense given the precision/recall trade-off and that removing stopwords gives us worse coverage, but could improve the quality of summary content.

5 Discussion

In certain LLR outputs, the chosen sentences may not constitute complete English sentences, necessitating more refined content selection methods. Moreover, there is room for improvement in the ordering of selected sentences to enhance cohesion and coherence. In future endeavors, our emphasis will be on refining information ordering and content realization. For instance, ensuring that pronouns align with the correct references will be a priority.

Our implementation and subsequent analysis of the SumBasic algorithm revealed that dialogues often dominate the top sentences selected for summaries, likely due to the high frequency of words like "said," "say," and "told." This observation suggests a nuanced challenge in the algorithm's ability to discern contextual importance, as dialogues, despite their length, disproportionately influence summarization. The application of tf-idf has shown promise in mitigating this issue, though the method's discounting formula requires further refinement. To address these challenges, we are exploring the integration of advanced preprocessing techniques such as lemmatization, stemming, and chunking. These methods aim to enhance our model's linguistic processing capabilities, potentially improving its ability to select more contextually relevant sentences and reduce the undue emphasis on dialogues, thereby aligning the summarization output closer to human-like comprehension and selection.

6 Conclusion

References

Lin, C.-Y. and Hovy, E. (2000). The automated acquisition of topic signatures for text summarization. In *Proceedings of COLING-2000*, page 495–501.

Louis, A. and Nenkova, A. (2013). Automatically Assessing Machine Summary Content Without a Gold Standard. *Computational Linguistics*, 39(2):267–300.

Nenkova, A. and Vanderwende, L. (2005). The impact of frequency on summarization. *Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005*, 101.

A Team Members and Workload Distribution

- Sheng Bi: (D2) Wrote `extract.py` script for extracting file names from XMLs. (D3) Responsible for SumBasic implementation and output.

- Long (Victor) Cheng: (D2) Wrote `{training | dev | eval}_process.py` scripts for formatting the original data files, getting documents from XML files and implementing preprocessing. Provided leadership to the team and monitored project progress. (D3) Led LLR implementation, managed D3.cmd and content_select.sh development, and maintained the Github repository to meet D3 requirements. Provided leadership to the team and monitored project progress.

- Catherine Wang: (D2) Made `.sh` scripts to get files given by the file name outputs from `extract.py`; wrote report. (D3) Responsible for evaluation (writing evaluation scripts and generating formatted ROUGE output and tables).

- Vicky Xiang: (D2) Conducted literature review for summarization task; handled project organization on GitHub; edited and reviewed `extract.py`; responsible for presenting in class. (D3) Responsible for evaluation (making table, discussion, and presentation).

- Carrie Yuan: (D2) Made presentation slides. (D3) Responsible for LexRank implementation and output.

B Additional software and data

Off-the-shelf tools we have used include:

- NLTK for tokenization

- `xml.etree.ElementTree` for analyzing XML files

- `rouge-score` package for calculating ROUGE scores