Summarization Project Report: Deliverable 4

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

Abstract

In this project, we explored three distinct approaches to accomplish the shared task of extractive summarization. The primary differentiation among the approaches lies in their methods of content selection: LLR (Log-Likelihood Ratio), sumBasic, and LexRank. Building upon the content selection methods, we devised three distinct strategies for information ordering: a Jaccard similarity-based baseline approach, entity-grid ordering, and chronological ordering. Additionally, we incorporated [mention the method/tool] for the final content realization. To evaluate the performance of the three approaches, we employed Rouge, InfoLM, and human evaluations. Overall, LexRank exhibited the best performance, whereas LLR and sumBasic yielded similar results.

1 Introduction

In the realm of text summarization, the quest for effective methods continues to drive research and development efforts. Summarization algorithms play a crucial role in distilling large volumes of text into concise and informative summaries, catering to the needs of various applications such as news aggregation, document summarization, and information retrieval.

In this project, we embark on a journey to explore and compare three distinct approaches to text summarization. The focal point of our investigation revolves around the methods of content selection, information ordering and content realization, all of which are pivotal components in the summarization process.

Our three approaches—LLR (Log-Likelihood Ratio), sumBasic, and LexRank—stand as pillars of innovation in the landscape of text summarization. Each approach offers a unique perspective and methodology for selecting and organizing content, thereby presenting diverse avenues for summarization excellence.

At the heart of our exploration lies the endeavor to elucidate the nuanced differences and comparative advantages of these approaches. We delve into the intricacies of content selection, examining how LLR, sumBasic, and LexRank leverage distinct algorithms and heuristics to identify salient information for inclusion in the summaries.

Furthermore, we unravel the complexities of information ordering, where the structuring of content holds paramount importance in ensuring coherence and readability. Through the lenses of Jaccard similarity-based baseline approach, entity-grid ordering, and chronological ordering, we navigate the labyrinth of ordering methodologies to unearth insights into their effectiveness and applicability.

content realization TODO

Additionally, our investigation extends beyond mere algorithmic evaluation, encompassing a comprehensive assessment of summarization quality. Leveraging metrics such as Rouge, InfoLM, and human evaluations, we endeavor to gauge the performance and efficacy of each approach, providing a holistic perspective on their summarization capabilities.

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.

Several enhancements were implemented to boost the performance of basic LLR. Firstly, a filtering mechanism was introduced to ensure that only grammatically correct English sentences are considered. This filtering process examines whether a potential sentence contains a subject, a verb, and entities, allowing selection only when all three conditions are met. Additionally, the all-MiniLM-L6-v2 model was employed to compute sentence embeddings. During the iteration over all valid sentences, sorted in reverse order by their weights, a threshold was set to exclude sentences similar to those already selected, as determined by cosine similarity. These improvements have resulted in more satisfying output summaries.

Furthermore, we avoided treating any hyper-parameters as arbitrary constants in our code. Instead, we treated them as command line arguments. For instance, in our methodology, we imposed a limit of 100 words or fewer for the length of the final summary, which is specified as a command line argument.

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 made improvements in preprocessing, such as filtering out sentences that are incomplete and lacks named entities, as well as removing words which have low tf-idf scores techniques.

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

- Preprocessing: Filter out sentences without proper subjects and verbs and lack named entities, as well as removing words which have low tf-idf scores such as say/said, etc.
- 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. (Our algorithm also takes care of the rare duplicates situation where identical high-score sentences from different files under the same document can persist.)
- Repeat until Done: Repeat the above process until the desired length is reached, ensuring diverse and relevant content coverage.

3.2.3 LexRank

The original paper (Erkan and Radev, 2011) introduce a graph-based method to compute relative importance of textual units. The main idea is that sentences "recommend" other similar sentences. The importance of a sentence depends on the importance of sentences "recommending" it. This makes intuitive sense that, the more similar a sentence to other important sentences in the cluster, the higher it ranks, and the more likely it is chosen as as summarization.

This method takes a mathematical approach to define the sentence salience, based on the concept of eigenvector centrality in a graph presentation of sentences. This method views the process of extractive summarization by identifying the most central sentences in a cluster. The first step is to calculate the similarity matrix for the sentences. There are multiple methods to compute the similarity between two sentences. The original paper uses tf-idf score as the sentence's vector representation, and use the cosine similarity as the element inside the similarity matrix. In Deliverable 4, we experimented with HuggingFace sentence transformer model (Wolf et al., 2019) to encode the sentences.

With the similarity matrix, we get a graph of n vertices, where n is number of sentences, and each vertex is connected to all the other vertices in the graph, since we have calculated their similarity before. Since we are only interested in significant similarities, we define a similarity threshold to remove the edges where the similarity between the two sentences is smaller than the threshold. The edges that are left in the graph represents the significant similarities that we are looking for. If we represent the current graph using a similarity matrix, we will have a matrix with only 0 and 1.

We normalize the matrix above by normalizing each column so that each column sum up to 1. Then we get a Markov transition matrix. Now we use

the power method to compute the stationary distribution for this Markov transition matrix. The intuition behind the stationary distribution is that, in the process of calculating the stationary distribution, the algorithm iteratively updates the importance scores (probability of visiting a sentence) based on the similarity between sentences and their current importance scores. This iterative process eventually leads to a stable distribution where sentences that are more similar to others tend to have higher probabilities of being visited, reflecting their importance or salience in the text. Finally we assign each sentence its LexRank score according to the final stationary distributions of the Markov transition matrix, and extract the sentences with the top LexRank score.

This method naturally ranks the sentences from the most important to the least important.

In summary, suppose we are interested in extracting the 5 most important sentences to use for our text summarization, our LexRank algorithm consists of the following steps:

- Preprocessing: Filter out sentences without proper subjects and verbs and lack named entities, as well as removing command stopwords. Filter out sentences with fewer than ten words in order to avoid skewed voting towards short sentences.

- Calculate Similarity Matrix: Calculate the pairwise cosine similarity between sentence vectors. The original papers uses tf-idf vector as the sentence representation.

- Thresholding the Similarity Matrix: Since we are only interested in "significant similarity", we will zero out the elements in the similarity matrix that is lower than the similarity threshold provided, and set the similarity that is larger than the significance threshold to 1.

- Calculate Degree Centrality: Calculate the degree for each nodes, which corresponds to number of significantly similar sentences.

- Calculate the final Markov Matrix: For each column inside the similarity matrix, divide it by its corresponding degree centrality, so that the sum of all elements in each column equal to 1, so that the similarity matrix is a Markov matrix.

- Apply power-method to compute the stationary distribution: Compute the stationary distribution of the markov matrix above. The stationary distribution corresponds to the centrality score of each sentences.

- Rank the sentences: Rank the sentences according to its centrality score.

- Get the summary: Take the 5 sentences with the largest centrality score.

3.3 Information Ordering

3.3.1 Jaccard Similarity-Based Baseline Approach

The baseline approach relies on Jaccard similarity and follows a straightforward implementation. Initially, we select the most heavily weighted sentence from the pool of selected sentences and place it as both the first and last sentence in our final summary, given that it's the only sentence at this stage. Subsequently, we iterate through the remaining sentences, seeking the most similar one to the last sentence in our summary, and append it accordingly. Notably, in computing the similarity between two sentences, we refrain from directly utilizing embeddings. Instead, inspired by (Chahal, 2016), we calculate the Jaccard similarity between the two sets of entities of two sentences to determine the similarity. Here's how the process works:

1.Entity Extraction: First, the entities (such as nouns, verbs, or other parts of speech) are extracted from each sentence using `spaCy`.

2.Set Formation: The entities extracted from each sentence are then organized into sets. Each set contains unique entities found in the respective sentence.

3.Jaccard Similarity Calculation: The Jaccard similarity between the two sets of entities is computed using the following formula:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Where: $|A \cap B|$ represents the size of the intersection of sets A and B (i.e., the number of entities common to both sentences). $|A \cup B|$ represents

the size of the union of sets A and B (i.e., the total number of unique entities in both sentences).

4.Interpretation: The resulting Jaccard similarity value indicates the degree of overlap or similarity between the sets of entities in the two sentences. A higher Jaccard similarity suggests a greater degree of similarity between the sentences in terms of the entities they contain.

By using Jaccard similarity based on sets of entities, the approach assesses the semantic similarity between sentences based on the concepts and entities they discuss, rather than relying solely on word-level or embedding-based comparisons.

3.3.2 Entity Grid

Barzilay et al. ((Barzilay and Lapata, 2008)) proposed an algorithm for analyzing sentence relationships, proving to be a robust tool for enhancing information ordering tasks.This technique focuses on identifying named entities within a document and tracking the evolution of their grammatical roles. Our adaptation retains the core methodology of their approach, showing promising enhancements in summary extraction and indicating potential for significant improvements with further refinements.

Our application of the entity-grid algorithm unfolds in five stages. Initially, we need to define the entity space. The original paper tried to solve the coreference problem by constructing entity classes that cluster named entities on the basis of their identity. We adopted a streamlined method compared to the original study. In our implementation, we simply utilized `spaCy.nlp` for sentence parsing, we extract named entities and apply a filter based on tf-idf scores, selecting tokens above the 90th percentile. This refined entity space forms the foundation for constructing an entity grid for each document in our dataset, which is then converted into a probability vector, creating our positive dataset.

Secondly, we need to do negative data generation. For that sake, we devise a straightforward method by creating inverted sentence order instances within each document, simulating coherence disruption.

Next, we compile probability vectors from both positive and negative instances forms our training matrix. It is worthwhile emphasizing that, to prevent collinearity in our linear model, we eliminate the $'-','-'$ feature pair due to its minimal grammatical content.

Then, We follow the original paper to run a supervised classification task using the support vector machine algorithm using `sklearn.SVM`. Finally, we use our SVM model for predictive analysis. The test data, comprising last week's extractive summary output, undergoes reordering to evaluate all permutations to make comparisons. This week, we selected the permutation that scored highest according to the `trained_model.decision_function()` method. We have been experimenting some other methods, such as evaluating subsequence block frequencies to evaluate whether our highest scored summary is consistent with the subsequence frequency distribution.

3.3.3 Chronological Ordering

One basic approach for info ordering is order by chronology. The idea is simple: given two extracted sentences, if they are from the same document, they should follow the original ordering in the document; if they are from different documents, the sentence in the document that has an earlier publication date ranked higher than the one with later publication date. This approach is clearly not ideal, but it at least realizes some level of ordering across sentences.

We implemented a greedy ordering algorithm described in the paper (Bollegala et al., 2012), using the idea described above as the preference function.

When implementing the choronological perference function, we extracted publication date from the date time within the document. For documents where such dateline is absent, we use the date time extracted from the document filename (1998/04/21 is extracted from the document titled with APW19990421.0284) as an approximate date time.

3.4 Content Realization

4 Results

4.1 Evaluation

In the development of our system, we employ automatic evaluation metrics to consistently evaluate the system's performance. For query-based tasks

Algorithm 1 Greedy Sentence Ordering Algorithm

procedure SENTENCEORDERING(X, PREFtotal(u, v, Q), where X is a set of unordered sentences extracted from the documents, PREFtotal(u, v, Q) is the preference function)
  $V \leftarrow X$
  $Q \leftarrow \emptyset$
  for each $v \in V$ **do**
    $\pi(v) \leftarrow \sum_{u \in V}$ PREFtotal$(v, u, Q) - \sum_{u \in V}$ PREFtotal(u, v, Q)
  while $V \neq \emptyset$ **do**
    $t \leftarrow \arg\max_{u \in V} \pi(u)$
    $\hat{\rho}(t) \leftarrow |V|$
    $V \leftarrow V - \{t\}$
    $Q \leftarrow Q \cup \{t\}$
    for each $v \in V$ **do**
      $\pi(v) \leftarrow \pi(v) +$ PREFtotal$(t, v, Q) -$ PREFtotal(v, t, Q)
  return $\hat{\rho}$

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).

In our evaluation process, we aim to capture the similarities between the system output and human-written reference summaries. Thus, we perform evaluation using automatic evaluation metrics, including ROUGE score and InfoLM, to systematically compare the system output to gold-standard references.

4.1.1 ROUGE score

We start with ROUGE to account for the number of summaries overlapping those in reference summaries. Using this metric, the overlapping n-grams is to be divided by the total number of n-grams in the reference summary and system summary respectively to calculate precision and recall scores. In this section, we present averaged ROUGE recall values, as a measure of the n-gram coverage provided by our system's output.

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 separately evaluate each model summary against our system output and produce an average ROUGE score over each evaluation, giving us a more comprehensive measure of the system's performance. 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 scores for each system. Table 2 presents the ROUGE recall scores for our improved systems after implementing enhancements such as filtering out ungrammatical or incomplete sentences and excluding highly similar sentences. To measure the improvement of each system, Tables 3 and 4 present the percent difference in ROUGE scores between the baseline and improved systems.

4.1.2 InfoLM

In addition, we adopted InfoLM metrics for the improved model after using information ordering techniques. An overview of InfoLM contains three main steps: 1. Mask each token in the input sentences (respectively from the target summaries and prediction summaries). 2. Using a weighted sum, the target summary distribution aggregation P and the prediction summary distribution aggregation Q are calculated as follows (Colombo et al., 2021):

$$P_{\boldsymbol{\Omega}|\mathbf{T}}(\cdot|\boldsymbol{x}; \theta; T) \sum_{k=1}^{M} \gamma_k \times p_\theta(\cdot|[\boldsymbol{x}]^k; \theta; T),$$

$$Q_{\boldsymbol{\Omega}|\mathbf{T}}(\cdot|\boldsymbol{y_i^s}; \theta; T) \sum_{k=1}^{N} \tilde{\gamma}_k \times p_\theta(\cdot|[\boldsymbol{y_i^s}]^k; \theta; T),$$

where γ_k and $\tilde{\gamma_k}$ are measures of the importance of the k-th token in the target and prediction summary respectively, x are the reference text and y are the prediction candidates, Ω denotes the vocabulary, T the temperature of the pre-trained language model, and θ the parameter of the pre-trained language model.

InfoLM consists of a family of untrained metrics designed for text summarization, with the possibility of loading pretrained language models. For the three content selection methods, we have calculated average the KL divergence, L1, L2 and L_{inf} distance scores across all 46 predictions doc-

Content Selection Method	ROUGE-1 Recall	ROUGE-2 Recall	ROUGE-L Recall
SumBasic (unigram with stopwords)	0.27048	0.04039	0.15626
SumBasic (unigram without stopwords)	0.34892	0.04348	0.17235
SumBasic (bigram with stopwords)	0.24920	0.03827	0.14567
LLR (with stopwords)	0.19683	0.02893	0.11047
LLR (without stopwords)	0.19562	0.02782	0.10829
LexRank	0.30984	0.05969	0.16391

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

Content Selection Method	ROUGE-1 Recall	ROUGE-2 Recall	ROUGE-L Recall
SumBasic	0.26561	0.03623	0.13823
LLR	0.24779	0.03813	0.13060
LexRank	0.39722	0.07308	0.19964

Table 2: ROUGE recall scores for the outputs of the improved content selection methods evaluated against human-written model summaries.

Content Selection Method	ROUGE-1 Recall	ROUGE-2 Recall
SumBasic	-23.8	-16.6
LLR	28.2	37.0
LexRank	26.7	22.4

Table 3: Percentage difference between ROUGE-1 and ROUGE-2 recall scores for each improved system's output and the corresponding baseline system's output.

Content Selection Method	ROUGE-1 Precision	ROUGE-2 Precision
SumBasic	45.8	60.2
LLR	19.2	27.7
LexRank	-12.4	-11.8

Table 4: Percentage difference between ROUGE-1 and ROUGE-2 precision scores for each improved system's output and the corresponding baseline system's output.

uments with their varied versions of human summaries, and we have recorded the results in Table 3. For KL divergence using LLR, document $D1007\text{-}A.M.100.B$ incurred an outlier value that is large, and is excluded from the average calculation. For KL divergence using SumBasic, both document $D1046A.M.100.H$ and $D1043A.M.100.H$ have their average scores excluded for the final average across all documents.

It can thus be seen that among the three content selection method, LexRank generated the KL divergence score closest to 0 across all documents, and there was no outlier to be excluded from the average calculation. LexRank also has the lowest L1, L2 and L_inf distance scores on average; even though SumBasic has average scores larger in magnitude for KL-divergence, L1 distance and L2 distance than those using LLR, it has an approximately same but slightly lower L_inf than that in LLR.

5 Discussion

5.1 Content Selection

In original LLR outputs, the chosen sentences may not constitute complete English sentences, which has been addressed by refined content selection methods. Moreover, efforts have been done in information ordering to enhance cohesion and coherence. From the current outputs we got some observations to consider: for example, when compared to model summaries typically comprising 5-7 sentences, the output summaries generated by LLR generally consist of 3-4 sentences, thereby reducing the significance of information ordering. Interestingly, despite normalizing weights during the calculation of each sentence's weight, the algorithm demonstrates a preference for longer sentences over shorter ones. In future endeavors, our emphasis will be on 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.

5.2 Evaluation

Table 2 demonstrates the success of our improved content selection algorithms, with the LLR method seeing a 26.7% increase in ROUGE-1 recall and the LexRank method seeing a 28.2% increase. Notably, the recall scores for the SumBasic method did not increase, but precision scores improved drastically, with over 45% improvement in ROUGE-1 precision.

We find the improved precision scores of the SumBasic and LLR methods to be quite reasonable. Our changes, including the removal of incomplete sentences, were primarily focused on improving the quality and readability of our generated summaries, rather than textual coverage. This could also have contributed to the negative change in recall scores for the new SumBasic model, as even though some of the sentences in the baseline model were ungrammatical, they may have had matching n-grams with some reference summaries.

We also note in our error analysis a bias towards longer sentences being selected. When a 100-word limit is imposed on summaries, when longer sentences are selected, fewer total sentences must be selected, which may negatively impact coverage. However, we do see that preventing overly similar sentences from being selected is highly beneficial to improving recall scores, as it prevents redundancy within the 100-word limit and allows for a broader coverage of the content within a succinct set of words. For the LLR system, we see that implementing a similarity threshold resulted in the greatest recall score improvement out of our three systems. On the other hand, for the SumBasic method, similar sentences are already discouraged from being selected repeatedly in the baseline method.

Content Selection Method	KL(exc. outliers)	L1	L2	L_inf
LLR	-5.276104444	0.4212043478	0.07614347826	0.03005869565
SumBasic	-5.852161364	0.43845	0.07721304348	0.02958913043
LexRank	-4.740382609	0.4140065217	0.0744673913	0.02884130435

Table 5: Average KL divergence, L1 distance, L2 distance and L_{inf} distance scores for the predicted summaries and human summaries across 46 document entries.

From a qualitative perspective, we can identify a few more errors that may contribute to our scores being lower than desired. We notice, for example, that datelines are frequently prepended to the first line of the original news documents, and those lines are often selected due to including important expository content. However, the inclusion of the dateline does not match with any of the human-written reference summaries, which usually exclude them. This could be resolved by splitting up these lines when processing the documents.

Finally, we continue to seek to improve the robustness of our system by continuing to improve the relevance and salience of selected content beyond simply including common phrases and frequently mentioned named entities. Highly common but inexact phrases can overweight sentences that otherwise involve very little salient information, though such an issue is difficult to quantify with string-matching metrics such as ROUGE score. In future iterations, we can deal with these issues by continuing to fine-tune content selection weights and relying on other evaluation metrics, such as Pyramid evaluation and human evaluation.

6 Conclusion

References

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

Bollegala, D., Okazaki, N., and Ishizuka, M. (2012). A preference learning approach to sentence ordering for multi-document summarization. *Information Sciences*, 217:78–95.

Chahal, M. (2016). Information retrieval using jaccard similarity coefficient. *International Journal of Computer Trends and Technology (IJCTT)*, 36.

Colombo, P., Clavel, C., and Piantanida, P. (2021). Infolm: A new metric to evaluate summarization & data2text generation. *CoRR*, abs/2112.01589.

Erkan, G. and Radev, D. R. (2011). Lexrank: Graph-based lexical centrality as salience in text summarization. *CoRR*, abs/1109.2128.

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.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. (2019). Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

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. (D4) Enhanced the content selection process of SumBasic and implemented the Entity Grid approach for information ordering.

- Long (Victor) Cheng: Provided leadership to the team and monitored project progress. (D2) Wrote `{training | dev | eval}_process.py` scripts for formatting the original data files, getting documents from XML files and implementing preprocessing. (D3) Led LLR implementation, managed D3.cmd and content_select.sh development, and maintained the Github repository to meet D3 requirements. (D4) Enhanced the content selection process of LLR and introduced a Jaccard similarity-based baseline approach for information ordering.

- 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). (D4) Performed ROUGE evaluation and comparison to baseline evaluation results, wrote discussion and qualitative error analysis.

- 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). (D4) Performed system-level re-examination on evaluation approaches, and implemented the InfoLM family of metrics which uses parameters from pretrained language model. Generated table to store all relevant InfoLM scores. Edited report and slides.

- Carrie Yuan: (D2) Made presentation slides. (D3) Responsible for LexRank implementation and output. (D4) Enhanced the content selection process of LexRank and implemented Chronological Ordering approach for information ordering.

B Additional software and data

Off-the-shelf tools we have used include:

- `nltk` for tokenization and stopwords
- `xml.etree.ElementTree` for analyzing XML files
- `rouge-score` for calculating ROUGE scores
- `scipy.spatial.distance` for calculating cosine similarity
- `sentence-transformers/all-MiniLM-L6-v2` model for getting embeddings of sentences
- `torch` and `transformers` for using LLMs
- `torchmetrics.functional.text.infolm` for evaluations on improved model with content selection