Summarization Project Final Report

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

Abstract

The task of multi-document extractive summarization has particular applications in the news domain, with the need for accurate, cohesive summaries that combine salient information from different sources. This project explores three distinct approaches to the task, utilizing the Log-Likelihood Ratio (LLR), SumBasic, and LexRank methods of content selection. Additionally, building upon these content selection methods, we devised three distinct strategies for information ordering: a Jaccard similarity-based baseline approach, entity-grid ordering, and chronological ordering. Finally, we incorporated shallow NLP processing methods and an LSTM-based approach for the final content realization. Using a combination of automatic evaluation metrics and human evaluation, we find that the LexRank algorithm produced the best performance, and discuss the impact of each component on our outcomes.

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.

This project explores and evaluates traditional methods of text summarization, focusing on a traditional approach that incorporates neural models at certain steps. The traditional approach has three main steps, which our project follows: content selection, information ordering, and content realization. Thus, content is extracted from the original documents and processed, rather than generated by our model.

The first step of the traditional approach to extractive summarization is content selection, or the identification of key content from the original text. We implement three content selection approaches: LLR, SumBasic, and LexRank. We present the results of ROUGE automatic evaluation of these systems, showing how different algorithmic and heuristic methods for identifying salient information across documents affects performance.

The next step is information ordering, or the determination of the most sound order in which to represent that information in the summary. We thus supplement each of our base content selection systems with information ordering methods including a Jaccard similarity-based approach, entity-grid ordering, and chronological ordering. Due to the complexities of logical information ordering that are difficult to capture via automatic string-matching evaluation metrics such as ROUGE score, we additionally incorporate InfoLM and human evaluation to provide insight into the effectiveness of each system.

For content realization, our approach involved the application of both a shallow NLP method and an LSTM-based approach. The shallow NLP method provided a quick and effective means of processing the text, while the LSTM-based approach offered deeper insights into the semantic structure. Following the application of these methods, we undertook additional refinements to further enhance the quality of the summaries. Leveraging the flexibility of regular expressions, we implemented a series of targeted improvements aimed at enhancing the coherence, clarity, and overall reader-friendliness of the summaries. By systematically applying these enhancements, we were able to refine the summaries to a level of quality that ensured a more engaging and informative reading experience for the audience.

2 System Overview

The summarization system is designed to condense text content into concise summaries through a multi-stage process. The system architecture con-

sists 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 news 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

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, the initial step is to tally 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 hyperparameters 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 eventu-

ally 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.

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 $\hat{\rho}$

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

3.4.1 Heuristic Approach

Based on the principles outlined in (Nenkova, 2008), we devised a heuristic approach to facilitate content realization. This approach was applied

to the summaries generated through our prior implementations of content selection and information ordering methodologies.

Initially, we leveraged the spacy package for coreference resolution. In processing each summary, we monitored the noun chunks and their corresponding heads. Upon encountering a new entity, we added its head to the set of coreferenced entities. Subsequently, when encountering another noun chunk with a head already present in the set, we replaced the chunk with its corresponding head. Since spacy noun_chunks do not encompass modifiers beyond the head noun, we scrutinized the subtree of the entire noun phrase to identify any modifiers. Removal of modifiers occurred only if the entity had been previously encountered. It is worth mentioning that our methods differ from (Nenkova, 2008) mainly in two aspects. Firstly, the rewrite of the coreferring noun phrases is embedded in the SumBasic content selection mechanism for the original paper, while we moved this procedure to the post-processing stage, so that this procedure can be applied to all our three content selection methods. Secondly, we restricted our coreferring noun phrases to named entities to avoid spurious or unpredictable rewrite; this is in contrast to the original paper, which applied the rewrite for all semantic content units which are by default carefully pre-selected salient terms.

Further refinements were made through heuristic enhancements, primarily employing regular expressions to enhance the coherence and readability of the final summaries. This involved eliminating sentence-initial adverbials and conjunct phrases up to the first comma. Additionally, we excised relative clause attributives and attributions lacking quotes. Finally, several substitutions were employed to restore word tokens into full sentences, thereby enhancing the summary's clarity and reader-friendliness.

3.4.2 Sentence Compression by Deletion with LSTM

Another approach we have attempted is based on the method in (Filippova et al., 2015), Existing compression systems heavily use syntactic information to minimize chances of introducing grammatical mistakes in the output. The main motivation behind this method is to design a sentence compression system that only uses tokens and has no access to syntactic or other linguistic information.

For dataset, we used 10000 sentences with labels (Filippova and Altun, 2013). Each sentence is tokenized and labeled with 1 (meaning this token will be kept in the compression) or 0 (meaning this token will not be kept in the compression). We used the usual 8:1:1 ratio for training, test and evaluation separation.

We opted to use Tensorflow for our LSTM implementation. The input, a vector of 258 dimensions, consists of embeddings of current word (256 dimensions) and the gold-standard label of the previous word (during training), or predicted label of previous word (during evaluation). We used the Google word2vec as the word embeddings. As suggested inside the (Filippova et al., 2015), we constructed a network consists of three stacked LSTM layer to allow the upper layers to learn higher-order representations of the input, interleaved with dropout layers to prevent overfitting. The model is trained to minimize the cross entry loss across predicted labels and gold-standard labels. Due to the time constraint, we opted to use a pretrained LSTM-based model from Hugging Face to compute the output from the sentences selected from previous steps directly.

For the current implementation, one limitation we have observed is the coherence. This system struggles with sentences with intervening commas, quotations, and other complex grammatical structures. In addition, since the buffer size for the max sentence length is 192 words, the system can be fed one sentence at a time, therefore the system removes components that it thinks irrelevant to the sentence itself, but are actually important in the context of multi-documents summarization.

4 Results

In the development of our system, we employ automatic evaluation metrics to consistently evaluate the system's performance. 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).

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 ROUGE score

We start with ROUGE score 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. We are particularly interested in recall score due to the fact that it provides insight into how well our system summaries cover the terms and tokens in the gold standard summaries.

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.

ROUGE score is typically a value between 0 and 1; all scores presented here have been multiplied by 100 for readability.

After implementing information ordering and content realization, Table 5 shows the final performance of each method over the dev set, and Table 6 shows the performance over the evaluation set. Based on these results, we also present an ablation study of SumBasic, our system with the best final recall score, and demonstrate the results of each component of the system in Table 7. We recognize that a string-matching measure like ROUGE score is known to have its limitations for evaluating logical ordering of sentences, so we also make use of human evaluation, which is discussed in the next section.

Although not every component improved the ROUGE recall scores of the output, we qualitatively observed an improvement in the outputs of our system after implementing filtering and content realization, in particular the quality and coherence of the summaries. We also note that the implementation of our content realization algorithms, in particular the ones for LLR and LexRank, may cause the tokens in our output summaries to further diverge from those in the reference summaries; however, we argue that this does not necessarily mean that the semantic content is worse, and in fact may represent the inclusion of topics not referred to in the reference summaries.

This brings to attention the need for other evaluation metrics, as well as human evaluation, which will be discussed in the following section.

4.2 InfoLM

In addition to ROUGE score, we adopt 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):

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

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

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

Content Selection Method	ROUGE-1 Recall	ROUGE-2 Recall	ROUGE-L Recall
SumBasic (unigram with stopwords)	27.05	4.04	15.64
SumBasic (unigram without stopwords)	34.89	4.35	17.23
SumBasic (bigram with stopwords)	24.92	3.83	14.57
LLR (with stopwords)	19.68	2.89	11.05
LLR (without stopwords)	19.56	02.78	10.83
LexRank	30.98	5.97	16.39

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

Content Selection Method	ROUGE-1 Recall	ROUGE-2 Recall	ROUGE-L Recall
SumBasic	26.56	3.62	13.82
LLR	24.78	3.81	13.06
LexRank	39.72	7.31	19.96

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.

Method	ROUGE-1 Recall	ROUGE-2 Recall
SumBasic	26.71	3.67
	(+0.7%)	(+13.8%)
LLR	23.91	3.66
	(-3.5%)	(-4.0%)
LexRank	21.2	3.36
	(-46.6%)	(-54.0%)

Table 5: ROUGE recall values for the outputs of each final system over the dev set, as well as the percent difference from the previous iteration.

Method	ROUGE-1 Recall	ROUGE-2 Recall
SumBasic	28.61	4.54
LLR	24.51	3.91
LexRank	23.11	4.28

Table 6: ROUGE recall values for the outputs of each final system over the evaluation set.

Iteration of method	ROUGE-1 Recall	ROUGE-2 Recall
Initial SumBasic (unigram)	27.0	4.0
Initial SumBasic (bigram)	24.9	3.8
With stopword removal	34.9	4.3
Improved method with filtering	26.5	3.6
Final method with content realization	26.7	3.7

Table 7: Ablation study of the SumBasic method, including ROUGE-1 and ROUGE-2 recall scores for each iteration of the method.

the three content selection methods, we have calculated average the KL divergence, L1, L2 and L_{inf} distance scores across all 46 predictions documents with their varied versions of human summaries, and we have recorded the results in Table 5. 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.

4.3 Pearson Correlation

The Pearson correlation coefficient is a statistic used to determine the strength and direction of a linear relationship between two variables. It ranges from -1 to +1, where -1 indicates a perfect negative linear relationship, 0 indicates no linear relationship, and +1 indicates a perfect positive linear relationship. This coefficient is suitable for evaluating summary correlation in Natural Language Processing (NLP) because it helps quantify the degree of association between different linguistic elements or features in text data. Researchers often use the Pearson correlation coefficient in NLP to analyze the semantic similarities between text summaries. (Schober et al., 2018). All corresponding results are shown in Table 10. For readability, scores are multiplied by 100.

4.4 Human Evaluation

To account for the issues with automatic evaluation, we implement human evaluation as part of our final evaluation process. Due to time and resource limitations, we evaluate over a subset of our outputs on the evaltest directory, consisting of 48 outputs from 11 document sets.

Our human evaluation process compares the outputs of our three final systems—LLR, SumBasic, and LexRank—with a baseline measure. The baseline measure is defined as the first n sentences from a randomly selected document from a given document set such that the total word count does not exceed 100 words. The baseline output thus consists of the first few sentences from a single random document in the document set.

In the human evaluation process, annotators receive the summaries blind; i.e., annotators have no knowledge of which summary has been produced by which system. Due to resource limitations, we performed an in-group human evaluation process, with the authors as annotators. As a result, authors may be somewhat familiar with the outputs of the system(s) they have implemented; however, annotators had strictly not seen these specific outputs in advance of the evaluations, and the ordering of outputs was randomized.

For each topic in the evaluation set, two human annotators were given access to the original documents to be summarized, as well as the four summaries—LLR, SumBasic, LexRank, and baseline—with no labeling of which system has generated which summary. Annotators were tasked with evaluating each summary along two axes: coverage and quality. Coverage referred to the ability of each summary to broadly cover important information from the documents without focusing on unnecessary details, which is comparable to ROUGE recall score. Quality referred to the coherence and written quality of the summary, including logical information ordering and grammatical En-

Content Selection Method	KL(exc. outliers)	L1	L2	L_inf
LLR	-5.28	0.42	0.08	0.03
SumBasic	-5.85	0.44	0.08	0.03
LexRank	-4.74	0.41	0.07	0.02

Table 8: Average KL divergence, L1 distance, L2 distance and L_{inf} distance scores for the predicted summaries and human summaries across 46 document entries generated in D4.

Content Selection Method	KL(exc. outliers)	L1	L2	L_inf
LLR	-5.25	0.42	0.08	0.037
SumBasic	-5.75	0.44	0.08	0.03
LexRank	-4.08	-0.42	-0.08	-0.03

Table 9: Average KL divergence, L1 distance, L2 distance and L_{inf} distance scores for the predicted summaries and human summaries across 46 document entries generated in D5. Blue statistics indicate those that witnessed an increasing trend in magnitude comparing to earlier InfoLM scores.

glish. Unlike coverage, it is difficult to approximate quality scores with string-matching techniques like ROUGE and InfoLM, as a good-quality summary can be very different from an equally good-quality reference summary.

Annotators' scores were averaged and normalized to a 0 to 1 scale to be comparable to ROUGE. The scores are shown in Table 11, multiplied by 10 for readability.

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.

After comparing the final outputs of the three approaches alongside the baseline approach, we observed that the summaries generated by LLR are shorter than those of the other methods. Upon examining the source code, we discovered that this is because, in the LLR approach, the content selection process halts once the current summary length plus the potential sentence length exceeds the upper bound of the summary length—a hyperparameter we set at the beginning of the project to 100 words. In such situations, this potential sentence is always discarded. Additionally, due to the LLR system's bias towards selecting longer sentences, the actual final summaries often fall short of the 100-word threshold. With less content in each summary, it is understandable why the coverage score of human evaluations for LLR appears to be lower than that of other methods, including the baseline. For future enhancements, the LLR method should aim to cover more content from the original news files to provide a more comprehensive summary. This could be achieved through the implementation of a more sophisticated content selection approach.

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

Content Selection Method	Pearson Correlation Score
LLR	16.47
SumBasic	16.68
LexRank	18.48

Table 10: Average Pearson correlation scores for devtest files.

Method	Human coverage score	Human quality score
Baseline	6.21	6.54
LLR	4.67	5.58
SumBasic	5.50	5.63
LexRank	5.17	4.50

Table 11: Human evaluation scores of each system compared to a baseline system.

and selection.

5.2 Content Realization

The heuristic method we implemented, while effective in reducing redundancy, exhibits several limitations. Notably, it inaccurately handles proper noun phrases such as "Gansu Province," replacing them with generic terms like "Province" upon subsequent mentions. This oversimplification presents a clear challenge, as the optimal replacement should reflect the specific context, suggesting alternatives like "this province" or "that province."

Another notable spurious situation that can result from our rule-based coreference class implementation is that elements appearing in the same head-noun class can actually represent different entities. In this situation, it does not make sense to use tf-idf as the only metric to choose which to present. For example, both "Another driver" and "a driver who worked with Roberts" are in the equivalent class of the head noun "driver", but they can represent different persons; for another example, both "the Tengger desert" and "the Badain Jaran desert" are in the same head-noun class of "desert"; also with both "Carrie Geddes" and "Connor Geddes" for the head-noun "Geddes", with both "rural China" and "northeast China" for the head-noun "China", etc. Our solution is to only include named entities that are proper nouns as our head-nouns to avoid such unpredictable entity mis-rewrite.

Furthermore, our approach to coreference resolution falls short in addressing the nuanced disambiguation of pronouns. While we successfully replaced noun phrases with their respective heads, we encountered difficulties in reconciling pronouns within the summary with their original contexts in the news. This issue underscores the need for more sophisticated techniques to ensure coherence and grammatical accuracy.

In particular, we believe that Large Language Models (LLMs) hold promise for improving coreference resolution compared to the shallow and heuristic methods employed in this project. LLMs, with their advanced capabilities in contextual understanding, are poised to offer more robust solutions to the challenges of coreference resolution, aligning summaries more effectively with the nuances of the original text.

5.3 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.

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.

In the final iteration of our system, we implement human evaluation to account for some of the issues of quality and information ordering not being robustly represented through automatic evaluation techniques. Our human evaluation results consist of coverage and quality scores; while coverage scores show a solid correlation with ROUGE recall scores, our quality scores also represent information that is not captured by string-matching evaluation techniques. Using these scores, we can see, for example, that though the LexRank technique with LSTM-based sentence compression results in good coverage of information through shorter sentences, it also results in a low quality score, with human annotators finding the compressed sentences to be overly simplistic and at times less clear. On the other hand, the shorter sentences allow for better information coverage within the same word count.

We note that the baseline metric performed extremely well and in fact outperformed our system outputs. This is a somewhat expected outcome, as this baseline is known to perform well for news articles, for which it is important that the first few sentences introduce the most important information and provide a thorough contextual overview. However, such a baseline may not perform as well over a different genre, such as summarizing a discussion or written literature. Therefore, we believe that the high coverage score of the baseline metric is expected, and it is still reasonable to consider the coverage scores of our system outputs as an important measure of the perceived quality of our system outputs by human readers.

Moreover, we argue that the quality measure for baseline outputs is largely arbitrary; because the baseline selects directly from the original documents with no content realization mechanics, it follows that the baseline will only be composed of well-formed sentences. Since the baseline only selects from the *beginning* of one specific document, the baseline sentences must logically follow from one another and not introduce any out-of-context named entities or pronouns, something that our content selection algorithms may be likely to do. This reinforces the need for us to continue to improve anaphora and coreference resolution in the output of our systems, which may be a route to improving the logical and written quality of system summaries. Looking ahead, the further adaptation of neural methods and large language models (LLMs) pose many opportunities for the improvement of information ordering and content realization.

6 Conclusion

In conclusion, our project delved into the realm of extractive summarization, exploring three distinct approaches to achieve this task. Through the utilization of LLR, sumBasic, and LexRank for content selection, coupled with strategies for information ordering including Jaccard similarity-based baseline, entity-grid ordering, and chronological ordering, we aimed to create comprehensive summaries. Our efforts culminated in the integration of both shallow NLP methods and LSTM-based approaches for content realization.

Evaluation using Rouge, InfoLM, and human assessments revealed LexRank as the top performer, while LLR and SumBasic demonstrated comparable results. Our project embraced a blend of traditional statistical methods and cutting-edge neural network approaches, underscoring our commitment to comprehensive exploration.

Looking ahead, we foresee the potential for LLMs to enhance extractive summarization tasks, particularly in the dynamic landscape of news. As we move forward, we remain poised to leverage emerging technologies and methodologies to further advance the field of extractive summarization.

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.

Filippova, K., Alfonseca, E., Colmenares, C. A., Kaiser, L., and Vinyals, O. (2015). Sentence compression by deletion with LSTMs. In Màrquez, L., Callison-Burch, C., and Su, J., editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 360–368, Lisbon, Portugal. Association for Computational Linguistics.

Filippova, K. and Altun, Y. (2013). Overcoming the lack of parallel data in sentence compression. In Yarowsky, D., Baldwin, T., Korhonen, A., Livescu, K., and Bethard, S., editors, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1481–1491, Seattle, Washington, USA. Association for Computational Linguistics.

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. (2008). Entity-driven rewrite for multi-document summarization. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*.

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

Schober, P., Boer, C., and Schwarte, L. A. (2018). Correlation coefficients: Appropriate use and interpretation. *Anesthesia and analgesia*, 126(5):1763—1768.

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

- Long (Victor) Cheng: Provided leadership to the team, monitored project progress and hosted weekly group meetings. Drafted Abstract, Introduction, System Overview, Conclusion and other parts of this Report he was responsible for. Maintained the Github repository. Developed Dx.cmd. (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 and developed content_select.sh. (D4) Enhanced the content selection process of LLR and introduced a Jaccard similarity-based baseline approach for information ordering. (D5) Produced baseline outputs for evaltest and collaborated with Sheng Bi to implement a shallow and heuristic approach for content realization.

- 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.

- Catherine Wang: (D2) Made `.sh` scripts to get files given by the file name outputs from `extract.py`; wrote preprocessing section of report. (D3) Wrote ROUGE evaluation scripts; analyzed and presented data in results section. (D4) Performed ROUGE evaluation and comparison to initial evaluation results. Carried out qualitative error analysis and wrote corresponding Discussion section. (D5) Designed and carried out human evaluation experiment and analyzed human evaluation results and ROUGE results. Created and formatted data tables; revised/edited Abstract, Introduction, System Overview, and Discussion sections of report.

- 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. (D5) Wrote scripts for calculating the pearson correlation coefficients across files in the devset. Added corresponding parts in the slides and report.

- 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. (D5) Implemented LSTM deletion-based sentence compression system.

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
- `spacy` to address coreference resolution and assess sentence legality
- `Tensorflow` to implement LSTM neural network
- `Sentence Transformers` to use pretrained model from Hugging Face