Graph-Based Unsupervised Summarization (GUSUM)

NLP : Final Project Paper

Kai-Hsuan Chan Yixuan Han Joshua Guillen

Introduction - Abstract

This project aims to enhance the GUSUM algorithm, an unsupervised extractive summarization model that utilizes graph-based techniques to identify and extract salient sentences from a document. GUSUM is designed to produce summaries without the need for labeled data, making it a powerful approach for large-scale summarization tasks.

Our research focuses on investigating modifications to GUSUM to enhance summarization quality. We plan to examine the impact of alternative sentence features, such as thematic relevance or more complex centrality metrics on the algorithm's performance. These features are scored using ROUGE, a model that compares generated summaries to human-expert-crafted summaries. Furthermore, we are exploring different similarity measures beyond cosine similarity, including alternative distance metrics (like forward and backward edges) to refine the sentence connections in the graph. These potential modifications were driven by the hypothesis that a more diversified set of features and similarity measures could improve GUSUM's ability to retain contextual relevance and coherence in the generated summaries, thus enhancing its accuracy in capturing the core essence of documents and thus producing higher ROUGE scores. Although certain features did indeed produce higher ROUGE scores, others did not perform as we had anticipated.

GUSUM - Features

The original GUSUM framework uses four main features (Tuba Gokhan, Phillip Smith, and Mark Lee, 2022 [1]):

1. Calculating sentence features for defining vertex weight.
2. Sentence embeddings by SentenceBERT to measure sentence similarities.
3. Graph creation by comparing all pairs of sentence embeddings obtained.
4. Ranking the sentences by their degree centrality in this graph using PageRank.

Calculating Sentence Features

GUSUM uses sentence features to determine the initial rank of the vertex in the graph instead of solely relying on vectorization. GUSUM focuses on four features for each sentence based on Shirwandhar and Kulkarni' standards (Nikhil S. Shirwandkar and Samidha Kulkarni, 2018 [6]). After scores for each sentence are determined, their sum is assigned as the weight of the vertex.

Sentence length

This feature is useful for filtering out unnecessary short sentences or phrases commonly found in articles, such as dates and author names. Short sentences do not contain much information and are typically not expected to be in a summary. To find the important sentence based on its length, the feature score is calculated using:

$$\text{Score}_{f1}(S_i) = \frac{\text{\# of Words in } S_i}{\text{\# of Words in Longest Sentence}}.$$

Sentence position

Sentence position is known to be relevant. The first and the last sentence of a document are usually important and contain a lot of information. Position feature scoring is calculated as follows (where N is the total number of sentences and P is the position of the current sentence):

$$\text{Score}_{f2}(S_i) = \begin{cases} 1 & \text{if it's the first or last sentence} \\ \frac{N-P}{N} & \text{otherwise} \end{cases}$$

Proper Nouns

Sentences with more proper nouns than the average sentence usually indicate that there is key information, making the sentence important when considering including it in the summary. This score calculates the ratio of the number of proper nouns in a sentence over the sentence length using POS tagging:

$$\text{Score}_{f3}(S_i) = \frac{\text{\# of Proper Nouns in } S_i}{\text{Length of } S_i}.$$

Tokenization (Numerical Tokens)

The presence of numerical tokens can indicate potentially important sentences to include in the summary. Numerical tokens are essentially the amount of numerical-related data found in a sentence. This is calculated using:

$$\text{Score}_{f4}(S_i) = \frac{\text{num-numeric}_i}{\text{Length}(S_i)}.$$

Sentence-BERT Embeddings

After sentences are extracted as well as their features, embeddings are produced for each sentence using Sentence-BERT, a network that uses Siamese and triplet network structures to

obtain semantically meaningful embeddings that will be able to be compared using similarity methods. Sentences are represented as fixed-sized vectors so that all sentences (including the source document) are mapped in the same semantic environment and viewed as inputs in the framework.

Graph Creation

The GUSUM framework represents a document as a graph where each node represents a sentence. Given a document D with sentences $(s_1, s_2, \ldots, s_n)$, we form a graph $G = (V, E)$ where V is the set of vertices corresponding to sentences s_i, and E is the edge set representing relationships between pairs of sentences v_i and v_j.These relationships are initialized primarily by calculating their centrality:

$$\text{Centrality}(S_i) = \sum_{j=1}^{N} e_{ij},$$

where e_{ij} represents the edge weight between sentences S_i and S_j.

After computing centrality scores, sentences are sorted in reverse order, and the highest scored sentences are included in the summary. The edges are then generated using cosine similarity to find semantic comparisons between sentences. Cosine similarity is defined as:

$$\text{Cosine Similarity} = \frac{\sum_{i=1}^{N} A_i B_i}{\sqrt{\sum_{i=1}^{N} A_i^2}\sqrt{\sum_{i=1}^{N} B_i^2}}.$$

Ranking and Summary Selection

After the graph is created, a ranking algorithm is used to find sentences that are most semantically similar to other sentences. This ranking algorithm defines importance as:

$$\text{Rank}(S_i) = v[i] \sum_{j=1}^{n} A[i, j]$$

where v_i is the corresponding feature score for sentence S_i and $A_{i,j}$ represents the adjacency matrix of the graph.

Finally, sentences are ranked and selected for the summary as:

$$\text{summary} = \text{topK}\{\text{Rank}(S_i) \mid i = 1, \ldots, n\}.$$

Dataset

The dataset used in our research was the CNN/DailyMail corpus, containing over 300,000 unique news articles sourced from CNN and the Daily Mail (Abigail See, CNN/DailyMail Dataset repository [7]). Although originally created for machine reading comprehension and abstractive question answering, this resource now supports both extractive and abstractive summarization tasks. To manage computational complexity, we focused on approximately 40,000 CNN articles (30,000 for training and 10,000 for testing). A key advantage of this dataset is that most articles are accompanied by human-expert summaries, enabling a more reliable and accurate evaluation of our generated summaries through ROUGE scoring.

Added Features in Enhanced GUSUM Framework

To enhance the GUSUM algorithm, we decided to look for factors that may help us improve ROUGE scores. ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics that are used to determine the quality of a summarization. It works by comparing generated summaries against reference summaries (which are typically human-produced). There are three types of ROUGE metrics to score these summaries: ROUGE 1, which calculates the overlap in individual words (unigrams) between the output and the reference summary; ROUGE-2, which is the same as ROUGE-1, but calculates overlap in bigrams between the output and reference summary; and ROUGE-L, which calculates the longest common subsequence between the output and reference summary. (Chin-Yew Lin, 2004 [4])

When evaluating important factors that could either improve or worsen the GUSUM model we looked at: Vectorization, Weight changes, Diversity, and TF-IDF (Keyword ranking). Through these features, we noticed that some factors drastically improved scores, while other features either plateaued results, had little to no improvements, or even decreased ROUGE scores.

Modifications to Existing Features

To enhance the GUSUM framework, we first thought to look at which of the original four features holds a greater importance in generating higher ROUGE scores. To figure this out, we decided to adjust the base-model vectorization weights on each individual feature so that their weight is higher than other features, giving them a higher prioritization. As shown in Figure 1, we observed that when Position and Numerical token weights were prioritized (individually), they yielded higher ROUGE scores. In addition, we aimed to identify sentences with the most diverse set of tokens, which allowed us to use centrality measures to calculate the relational properties among these sentences and generate a concise summary. To achieve this, we excluded sentences with a cosine similarly exceeding 0.8, leaving us with a set of diverse sentences. In modifying the diversity of the sentences that are extracted, we saw minor improvements in the GUSUM model (as shown in Figure 1).

After initially ranking sentences based on their importance scores, we iteratively select the top-ranked sentences that do not exceed a specified similarity threshold with any previously chosen sentence. This ensures that the final summary contains more varied and thematically distinct sentences.

In more formal terms, let S be the set of candidate sentences, each associated with an embedding $\mathbf{e}(s)$. We aim to construct a subset $T \subseteq S$ of top k sentences such that no two selected sentences are too similar. Formally:

$$\text{If } s \in S \text{ is to be selected, then } \forall t \in T, \quad \text{sim}(\mathbf{e}(s), \mathbf{e}(t)) < \alpha,$$

where $\text{sim}(\cdot, \cdot)$ represents the cosine similarity function and α is our similarity threshold (which is a score of 0.8).

In the code, each sentence is first scored, and then we choose sentences in descending order of score, adding a new sentence only if its cosine similarity with all previously chosen sentences is below the threshold α.

By implementing this approach, we ensure that the final summary is not only high-scoring according to the given features but also more diverse, thereby potentially capturing a broader range of key information from the source text.

Keyword Ranking Through TF-IDF

Another feature that was implemented to enhance the GUSUM framework was Keyword ranking using Term Frequency-Inverse Document Frequency (TF-IDF). TF-IDF is a statistical metric that is used to evaluate the importance of a word in a document relative to a collection of other documents (a corpus). A word with a high TF-IDF score is considered to be more significant within the document. To leverage TF-IDF effectively, we decided it would be best if we integrated it as an additional feature alongside the original base model and if we trained it on our dataset (which consists of about 30k CNN articles for training and about 10k articles for testing). More specifically, TF-IDF was integrated by multiplying the base scores by the tf-idf score, which resulted in a slight improvement in ROUGE scores as shown in Figure 1.

Figure 1: ROUGE Scores on Different Features and Diverse Model

Model	ROUGE-1	ROUGE-2	ROUGEL	Length_W	Position_W	PPN_W	Num_W	TFIDF_W
base	0.346502	0.132939	0.408088	1	1	1	1	0
base_diverse	0.349789	0.134664	0.409511	1	1	1	1	0
len	0.306688	0.103654	0.382694	5	1	1	1	1
pos	0.368128	0.143984	0.426085	1	5	1	1	1
ppn	0.335283	0.125632	0.415006	1	1	5	1	1
num	0.343889	0.132629	0.405385	1	1	1	5	1
tfidf	0.349337	0.134458	0.41048	1	1	1	1	1

Other Implementations

We also considered implementing a graph-based keyword extraction approach inspired by TextRank (Rada Mihalcea and Paul Tarau, 2004, [9]). Unlike sentence-level vertices, this method involves representing individual lexical units (like nouns or adjectives) as vertices in a graph. Edges are drawn between these vertices if their corresponding words co-occur within a specified window of text. By applying a ranking algorithm similar to TextRank on the graph, each unit is assigned a score that reflects its importance in the document. After convergence, highly ranked words are selected as keywords, and sequences of adjacent keywords are merged into multi-word keyphrases. Although this approach has shown promising results in keyword extraction tasks, we were unable to fully implement and evaluate it within our current framework due to time constraints, and thus have excluded it from the scope of this research.

Centrality

In the original GUSUM framework, centrality is calculated as the sum of all cosine similarities between a sentence and its connected neighbors. While this captures a notion of centrality, it does not account for the positional importance of sentences (especially in news articles), such as the influence of initial sentences in a paragraph.

To address this, we introduced a modified centrality metric that incorporates positional

weighting and directionality. Forward edges represent sentences that come after the extracted sentence, while backward edges represent sentences that come before it. We assigned backward edges a higher priority for scoring initial sentences in a paragraph. Less important sentences, like the final ones, have fewer backward edges and thus less prioritization. This encourages the selection of sentences well-supported by information introduced earlier. (Hao Zheng and Mirella Lapata, 2019, [8])

The new centrality is calculated as:

$$\text{Centrality}(S_i) = \lambda_1 \sum_{j<i} e_{ij} + \lambda_2 \sum_{j>i} e_{ij}.$$

Where λ_1 represents backward edges and λ_2 represents forward edges of $e \in E$. After applying this approach, we confirmed its effectiveness. As shown in Figure 2, increasing the weight for backward edges increased ROUGE scores, with the highest ROUGE scores achieved by the `base_diverse` GUSUM model.

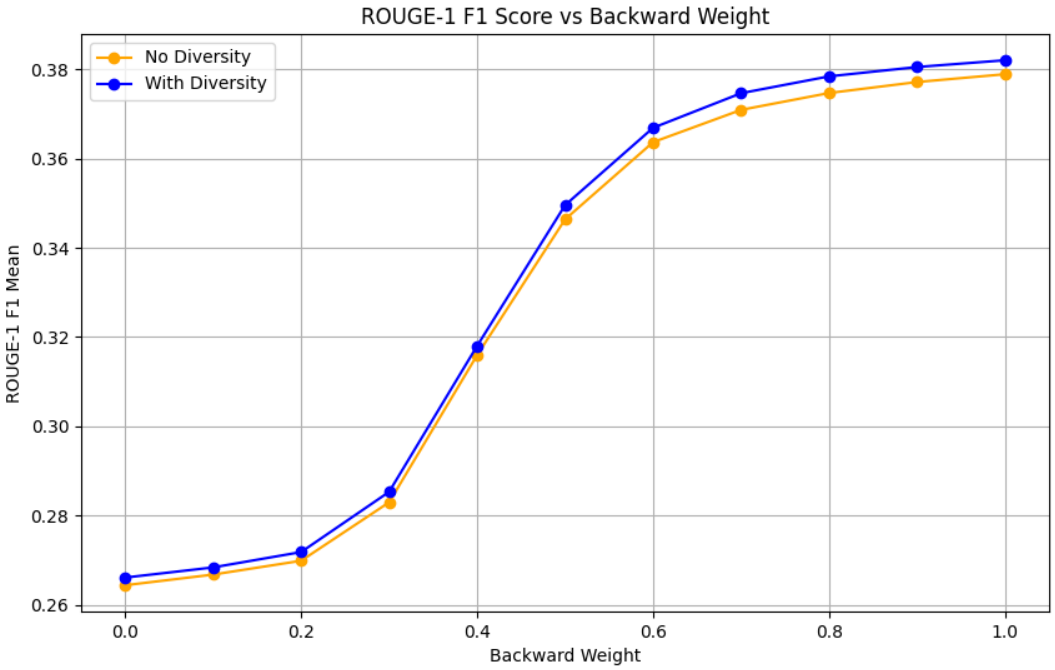

Figure 2: Graph of ROUGE-1 (with and without diverse base) F1 scores relative to their Backward weight.

Conclusion

In this research, we enhanced the GUSUM framework by introducing modifications to its features and graph-based centrality approach, ultimately improving the quality of generated summaries as demonstrated by ROUGE metrics. First, we refined the structure of centrality by incorporating both forward and backward edges, emphasizing the importance of sentences that are well-supported by earlier text (especially with the first sentences of a document/paragraph). This directional weighting more accurately extracted the overall thematic sentiment within the document, thereby enhancing sentence selection. We also integrated a TF-IDF keyword ranking feature, which provided an additional dimension for identifying salient sentences. Additionally, we increased diversity in selected sentences by excluding those with high similarity scores. This approach ensured that the final summary contained a richer variety of information, further improving ROUGE performance and thus

producing a higher quality summary. Taken together, these enhancements demonstrate the potential of graph-based strategies for producing more coherent and contextually representative summaries. Our findings suggest that these modifications can lead to more effective unsupervised extractive summarization approaches, opening pathways for future research on integrating additional semantic features into GUSUM.

References

[1] Tuba Gokhan, Phillip Smith, and Mark Lee. (2022). *GUSUM: Graph-based Unsupervised Summarization Using Sentence Features Scoring and Sentence-BERT*. In Proceedings of TextGraphs-16: Graph-based Methods for Natural Language Processing, pages 44–53, Gyeongju, Republic of Korea. Association for Computational Linguistics.

[2] Hao Zheng and Mirella Lapata. (2019). *Sentence Centrality Revisited for Unsupervised Summarization*. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 6236–6247, Florence, Italy. Association for Computational Linguistics.

[3] Jun-Ping Ng and Viktoria Abrecht. (2015). *Better Summarization Evaluation with Word Embeddings for ROUGE*. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1925–1930, Lisbon, Portugal. Association for Computational Linguistics.

[4] Chin-Yew Lin. (2004). *ROUGE: A Package for Automatic Evaluation of Summaries*. In Text Summarization Branches Out, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

[5] Zhihao Zhang, Xinnian Liang, Yuan Zuo, Zhoujun Li. (2023). *Unsupervised Abstractive Summarization via Sentence Rewriting*. Computer Speech & Language.

[6] Nikhil S. Shirwandkar and Samidha Kulkarni. (2018). *Extractive Text Summarization Using Deep Learning*. In 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), pages 1–5.

[7] Abigail See. (n.d.). *CNN/DailyMail Dataset*. Homepage: `https://huggingface.co/datasets/abisee/cnn_dailymail` Repository: *CNN/DailyMail Dataset repository*

[8] Hao Zheng and Mirella Lapata. (2019). *Sentence Centrality Revisited for Unsupervised Summarization*. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 6236–6247. Available at: `https://aclanthology.org/P19-1628.pdf`.

[9] Rada Mihalcea and Paul Tarau. (2004). *TextRank: Bringing Order into Texts*. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pages 404–411. Available at: `https://aclanthology.org/W04-3252.pdf`.