Knowledge and Information Systems (2024) 66:7557–7580
https://doi.org/10.1007/s10115-024-02195-3

REGULAR PAPER

GoSum: extractive summarization of long documents by reinforcement learning and graph-organized discourse state

Junyi Bian[1,8] · Xiaodi Huang[2] · Hong Zhou[3] · Tianyang Huang[4] · Shanfeng Zhu[4,5,6,7,8]

Received: 17 March 2024 / Revised: 2 July 2024 / Accepted: 23 July 2024 /
Published online: 22 August 2024

Abstract
Summarizing extensive documents involves selecting sentences, with the organizational structure of document sections playing a pivotal role. However, effectively utilizing discourse information for summary generation poses a significant challenge, especially given the inconsistency between training and evaluation in extractive summarization. In this paper, we introduce GoSum, a novel extractive summarizer that integrates a graph-based model with reinforcement learning techniques to summarize long documents. Specifically, GoSum utilizes a graph neural network to encode sentence states, constructing a heterogeneous graph that represents each document at various discourse levels. The edges of this graph capture hierarchical relationships between different document sections. Furthermore, GoSum incorporates offline reinforcement learning, enabling the model to receive ROUGE score feedback on diverse training samples, thereby enhancing the quality of summary generation. On the two scientific article datasets PubMed and arXiv, GoSum achieved the highest performance among extractive models. Particularly on the PubMed dataset, GoSum outperformed other models with ROUGE-1 and ROUGE-L scores surpassing by 0.45 and 0.26, respectively.

Keywords Extractive summarization · Graph neural network · Reinforcement learning · Long document summarization

1 Introduction

Document summarization aims to generate concise and informative summaries from given texts. This enables readers to comprehend the main points and findings of a research paper without needing to read the entire document. There are two primary categories of approaches used in summarization: extractive and abstractive. Extractive approaches involve scoring and filtering sentences from a given document. In contrast, abstractive approaches typically employ a sequence-to-sequence (seq2seq) framework to generate summaries.

In recent years, abstractive summarization methods have progressively demonstrated remarkable performance [1, 2], particularly with the emergence of large language models [3–7], which can excel in evaluation metrics even without fine-tuning. However, a drawback of abstractive approaches is their tendency to occasionally generate meaningless and unfaithful summaries, as observed in [8, 9]. Particular, concerning is the so-called hallucination issue

Extended author information available on the last page of the article

in large language models. Despite the smooth flow and high ROUGE [10] scores of the generated summaries, there can exist a semantic gap between them and the gold standard summaries. In contrast, extractive summarization models directly extract sentences from the original text, ensuring complete consistency with the original expression. Another advantage of extractive summarization is its significantly lower computational resource requirements compared to abstractive approaches.

In this paper, we concentrate on the utilization of extractive models for summarizing scientific literature. While extractive summarization [11, 12] has received extensive attention within short summarization datasets such as CNN/DailyMail [13] or Wikihow [14], progress in long text summarization has been comparatively sluggish. This is primarily attributed to two key challenges. Firstly, the increasing length of input documents escalates the memory requirements of the model, particularly those employing transformer [15] architectures. Secondly, the intricate discourse structure inherent in long-form documents needs to be effectively incorporated. When reading lengthy texts, especially scientific literature, readers often rely on the discourse structure to obtain a comprehensive overview of the entire text. For instance, when readers seek to grasp content regarding methodologies, they typically turn to the "methodology" section, and to learn about experimental results, they refer to the "experiments" section. This discourse information plays a crucial role in comprehending the meaning and context of individual sentences within the text. In the context of extractive summarization, leveraging this structural information can significantly enhance the sentence representation. Previous methods that encode sentences and sections separately face challenges in capturing the hierarchical structure of the document. The lack of integration between these two levels of granularity hampers the model's capacity to fully grasp and represent the hierarchical relationships within the document.

To overcome this limitation, we propose leveraging a graph neural network (GNN) to capture the hierarchical structure of a document. For graph construction, we treat each sentence and section as a node within a heterogeneous graph. After the message passing through the graph, the final embedding of each sentence node can encapsulate the discourse information brought by the input's section structure. Using such sentence representations for summary sentence extraction effectively utilizes the input's section structural information. Additionally, the GNN architecture introduced in this paper exhibits a computational complexity that scales linearly with the number of input sentences, enabling it to effectively process longer inputs.

Unlike abstractive approaches, which are trained directly using available gold summaries, extractive models rely on a search algorithm, often utilizing greedy search, to generate oracle training labels. Traditional extractive approaches employ cross-entropy loss during training to maximize the probability of generating the oracle label. However, this training approach encounters several issues: (1) Greedy search results in a single suboptimal oracle label for each document-summary pair. The limited availability of such positive pairs can lead to underfitting [16]. (2) There exists a mismatch between the learning objective and the evaluation criterion, such as the ROUGE metric.

To address the challenges encountered during the training of extractive summarization models, we have employed reinforcement learning techniques. In particular, we have adopted offline sampling by employing beam search to construct the top-k oracle labels. This approach of sampling multiple oracle labels during training effectively mitigates the problem of having insufficient training samples. Furthermore, for each oracle label generated, a reward is calculated using the ROUGE metric to address the second issue.

In summary, the main contributions of this paper are as follows:

- GoSum: We introduce GoSum,[1] a novel graph-based discourse-aware extractive summarization model. Operating on a subsentential discourse unit level, GoSum generates concise and informative summaries.
- Integration of Reinforcement Learning with GNN: We effectively integrate reinforcement learning with GNN within GoSum. By obtaining sufficient samples in reinforcement learning. GoSum utilizes GNN to capture discourse information, specifically the discourse hierarchy, for extracting compact summaries.
- Experimental Validation: We conduct comprehensive experiments to validate GoSum's performance. It achieves state-of-the-art results among extractive baselines on two benchmark datasets: PubMed and arXiv. These results demonstrate the effectiveness of our proposed approach in summarizing scientific literature.

The organization of this paper is as follows. We review related work in Sect. 2. The method is presented in Sect. 3. Section 4 reports the experimental results, including ablation studies, human evaluation, and case study. Section 5 concludes this paper.

2 Related work

2.1 Long document summarization

In contrast to the success of BERT-based models in short-input summarization [11, 17, 18], long document summarization faces challenges due to lengthy input sequences. Abstractive models have predominantly focused on exploring transformer architectures to handle entire long inputs. For instance, BigBird [19] substitutes full self-attention with random attention, global attention, and window attention. A similar architecture, Longformer [20], proposes dilated sliding window attention. Additionally, BART-LS [1]adopts pooling-augmented blockwise attention, while HEPOS [21] utilizes head-wise positional strides attention to handle longer text inputs.

In contrast, most extractive models [22–24] extract summaries by encoding sentences independently while simultaneously considering the global or discourse information of the document. Some extractive methods [25, 26] also adopt Longformer [20] to encode all sentences simultaneously for sentence extraction. The GoSum proposed in this paper also falls under this category.

2.2 Graph-based extractive summarization

Early graph-based unsupervised methods [27] rely on explicit surface features. These methods construct a similarity graph between sentences and treat extractive summarization as a node ranking task. In recent years, researchers have shifted toward supervised summarization using graph neural networks. HSG [28] is the first to propose a heterogeneous graph neural network for extractive document summarization. HahSum [29] addresses inter-sentence redundancy in graph construction. Topic-GraphSum [30] introduces high-level semantic nodes in a graph as an intermediate to bridge long-distance sentences. HEROS [24] extends the graph-based approach to the long texts by incorporating the discourse information from the input articles.

[1] The source code of GoSum is available at https://github.com/Eulring/GoSum.

These methods treat sentences and words as nodes in a graph, while DiscoSum [31] uses a graph based on RST trees to capture the long-range dependencies among discourse units. The methods mentioned above represent both words and sentences as nodes in their graphs. In contrast, GoSum uses a more streamlined approach, focusing solely on sentences and sections as nodes in its graph structure.

2.3 Reinforcement learning based summarization

Wu and Hu [32] consider the fluency of the summary during extraction and include fluency scoring in the reward function alongside the maximization of the ROUGE metric, marking one of the earliest applications of reinforcement learning in extractive summarization. Some research has found that ROUGE-based evaluation metrics do not always align with readers' preferences. Böhm et al. [33] developed a reward function for summarization by manually scoring the quality of summaries, learning from human judgments. Similar efforts include [34], which also used manually designed reward functions to simulate readers' evaluations of summary content.

In another approach, Chen and Bansal [35] enhanced the efficiency of generative methods by first selecting important sentences and then generating summaries, integrating selection and generation through policy gradient methods for training. Local–Global+RdLoss [36] represents an improved version of the Local-Global [22] method, introducing reinforcement learning to address sentence redundancy. In contrast, MemSum [37] incorporates extraction history [12] into the Markov decision process for better summary extraction. The paper by Pang and He [38] was among the pioneers in employing offline reinforcement learning in the domain of abstractive summarization. Following this, OREO [39] utilized offline reinforcement learning in extractive summarization by pre-sampling multiple high-quality training samples. Both GoSum and MemSum [37] treat the task of sentence extractions as a Markov decision process. Additionally, GoSum uses offline reinforcement learning during its training process, drawing inspiration from OREO [39],

3 Method

Figure 1 illustrates the architecture of GoSum. Given a document and its section structure as input, GoSum constructs a graph to model the input. During the sentence extraction phase, GoSum sequentially extracts sentences, incorporating three types of embeddings for each sentence: (1) discourse awareness embedding, (2) global context embedding, and (3) extraction history embedding.

3.1 Task definition

Extractive summarization involves selecting a subset of sentences from a document to create a concise summary. Let's denote a document as $D = \{s_1, s_2, ..., s_n\}$ where n represents the number of sentences in the document. The goal of the extractive summarizer is to generate a sequence of indexes $\bar{X} = \{\bar{x}_1, \bar{x}_2, ..., \bar{x}_m\}$ that determine which sentences from input should be included as summary. Here, $\bar{x}_i$ denotes the index of a sentence in the document D. The combination of sentences$\{s_{\bar{x}_1}, s_{\bar{x}_2}...\}$ constitutes the extractive model's output $\bar{Y}$.

Summarization datasets only provide the input document D and the corresponding summary Y, without offering labels $\bar{X}$ that can be used for extractive summarization training.

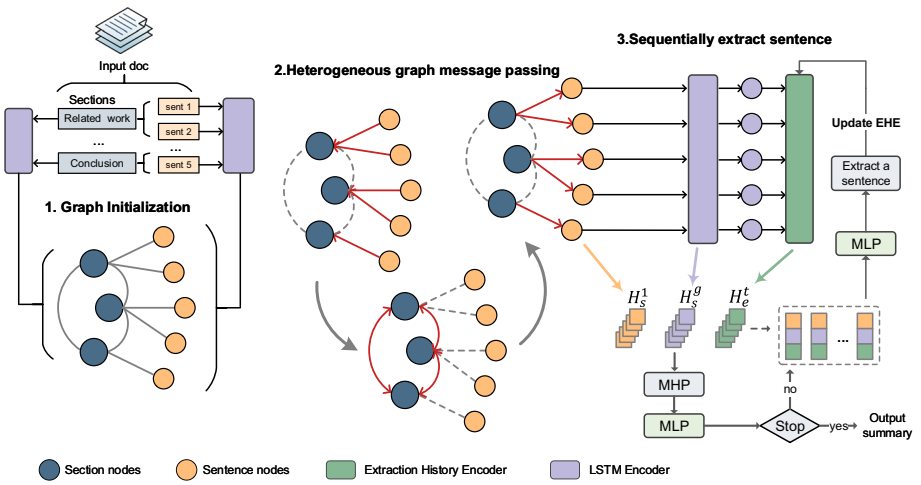


Fig. 1 The overall framework of GoSum. MHP: multi-head pooling, and MLP: multilayer perceptrons

Generally, extractive summarization methods employ a search algorithm (usually greedy search or beam search) to construct oracle labels $\hat{X} = \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_m\}$ based on $< D, Y >$.

3.2 Extractive summarizer by policy gradient

The proposed GoSum model can be regarded as an agent. At time step t, the state of the agent is $S_t = (D, \bar{x}_{1:t-1})$, indicating that $t-1$ sentences have been extracted from document D. Upon completion of the extraction, the agent receives a reward r to assess the performance of the extraction. This paper adopts a policy gradient approach to optimize GoSum. The so-called policy can be viewed as the probability of taking action A(selecting a sentence) at time t given the state S_t and under the parameters θ of the policy network:

$$\pi(A|S_t, \theta) = Pr(\bar{x}_t = A|S_t, \theta) \tag{1}$$

$$= Pr(\bar{x}_t = A|D, \bar{x}_{1:t-1}, \theta) \tag{2}$$

Let $S_0 = D$ represent the initial state, indicating that no sentences have been extracted yet. $v_{\pi_\theta}(S_0)$ denotes the expected value function starting from the initial state under the policy π_θ. Thus, the objective of reinforcement learning can be transformed into minimizing negative form of the initial state value function $v_{\pi_\theta}(S_0)$.

$$\mathcal{L}(\theta) = -v_{\pi_\theta}(S_0) \tag{3}$$

Additionally, the reward at time t is given by $R_t = \sum_{i=t} \gamma^{i-t} r_i$, where γ is the discount factor. During the reinforcement learning, the parameters θ of the policy network are continuously adjusted to minimize $\mathcal{L}(\theta)$:

$$\nabla\mathcal{L}(\theta) = -\sum_{t=1} \gamma^t Pr(S_t|S_0, \pi_\theta) \sum_A q_{\pi_\theta}(S_t, A) \nabla\pi_\theta(A|S_t) \tag{4}$$

The $q_{\pi_\theta}(S, A)$ denotes the action-value function, representing the reward for taking action A in state S under policy π_θ. Given the large state space of S, the Monte Carlo sampling method is commonly used to approximate the gradient. Consequently, the optimization

objective function can be written as follows:

$$\nabla \mathcal{L}(\theta) = -E_{\hat{A}_t, \hat{S}_t \sim \pi} \left[\gamma^t r_t \nabla_\theta \log \pi(\hat{A}_t | \hat{S}_t, \theta) \right] \tag{5}$$

where $\hat{A}_t$, $\hat{S}_t$ are obtained through sampling, and r_t is the corresponding reward. Skipping some proofs according to Williams [40] and setting $\gamma = 1$, the simplified formula for parameter updates can be derived as follows:

$$\theta_{t+1} \leftarrow \theta_t + \alpha r_t \nabla \log \pi(\hat{A}_t | \hat{S}_t, \theta) \tag{6}$$

where α represents the learning rate during training. Additionally, the setup of the reward function is crucial in reinforcement learning. This paper adopts the most commonly used reward value design in extractive summarization, where r is calculated based on the ROUGE score between the sampled summary $\hat{Y}$ and the actual summary Y.

$$r = \frac{1}{3} \left(\text{ROUGE-1}_f(\hat{Y}, Y) + \text{ROUGE-2}_f(\hat{Y}, Y) + \text{ROUGE-L}_f(\hat{Y}, Y) \right) \tag{7}$$

Typically, extractive summarization models extract a fixed number of sentences as the summary. However, by adopting a stopping mechanism, it is possible to dynamically determine when to stop the extraction process. Therefore, the policy likelihood can be expressed as follows:

$$\pi(A_t | S_t, \theta) = p(\text{stop} | S_t, \theta) p(A_t | \text{stop}, S_t, \theta) \tag{8}$$

At each step, the policy network first outputs a probability p_{stop}. If p_{stop} exceeds a predetermined threshold, the model will stop the extraction process. Otherwise, it will continue to select the next sentence for the summary. To prevent the model from extracting too many sentences, this paper sets a maximum limit of extracted sentences, k_{max}, for different datasets.

As indicated by the formula, an important aspect of the policy gradient method is obtaining the state S_t. In GoSum, the state S_t consists of three types of feature embeddings: (1) discourse awareness sentence embedding, (2) global contextual embedding, and (3) extraction history embedding. At time t (after t sentences have been extracted), GoSum generates these three types of embeddings for each sentence that has not yet been extracted. Next, we will detail the calculation for these three types of embeddings.

3.3 State encoder

As depicted in Eq. (8), the extractor determines when to stop the extraction process or compute the score for each remaining sentence based on the state S_t. This state is formed by concatenating three types of vectors:

- Discourse Awareness Representation $H_{\text{discourse}}$: This vector represents the discourse structure information of the sentence within the paper's context, capturing its structural information. (in Sect. 3.3.1)
- Global Context Representation H_{global}: This vector represents the sentence representation in relation to the global context of the document. (in Sect. 3.3.2)
- Extraction History Embedding H_{history}: This vector characterizes the attributes of the results already extracted during the summary sentence extraction process, aiming to avoid redundancy in the extracted sentences. (in Sect. 3.3.3)

$$S_t = [H_{discourse}; H_{global}; H_{history}] \tag{9}$$

3.3.1 Graph-based discourse awareness encoder

Graph construction: GoSum utilizes a heterogeneous graph to represent sections and sentences of a document at the discourse level. The graph consists of two kinds of nodes: sentence nodes and section nodes. In contrast to previous graph-based summarizers [24, 28–30], GoSum does not include word nodes in the graph construction. There are two reasons for excluding word nodes. First, introducing word nodes would significantly increase the size of the graph, thereby increasing the model's memory usage and runtime. Second, the information conveyed by word nodes to sentence nodes primarily pertains to the representation of the sentence's local content. Consequently, the use of a simple encoder such as LSTM is sufficient for processing this information.

In the graph of GoSum, edges are established between each sentence and the section that contains the sentence. Furthermore, a fully connected subgraph is created within each section, allowing information to flow between sections.

Graph initialization: Once the graph is constructed, we assign an initial representation to each node. Suppose that a sentence in a document consists of $|s|$ words: $(sw_1, sw_2, ..., sw_{|s|})$. Similarly, the text of a section, such as "Related work," is composed of $|c|$ words: $(cw_1, cw_2, ..., cw_{|c|})$. To create the initial representations, GoSum first employs Glove [41] word embeddings to embed these words. Then, it uses BiLSTM [42] with multi-head pooling (MHP) to generate sentence representation h_s^0 and section representation h_c^0:

$$h_c^0 = \text{MHP}\left(\text{BiLSTM}(\text{Glove}(cw_1, cw_2, ..., cw_c))\right) \tag{10}$$

$$h_s^0 = \text{MHP}\left(\text{BiLSTM}(\text{Glove}(sw_1, sw_2, ..., sw_s))\right) \tag{11}$$

Graph attention networks: Next, GoSum uses a graph attention layer (GAT) [43] to update the semantic nodes given the available graph G and its initial node features h_s, h_c. The expressions for GAT are as follows:

$$e_{ij} = \text{LeakyRELU}(W_a[W_q h_i; W_k h_j]) \tag{12}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})} \tag{13}$$

$$h_i' = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W_v h_i\right) + h_i \tag{14}$$

Here, W_a, W_q, W_k, and W_v are trainable weights, and h_i is the node representation of the ith node in the graph. $\mathcal{N}_i$ is the neighbor nodes of node i.

Message passing: GoSum utilizes three rounds of message passing to update the representation of each node, which are: (1) from sentence to section, (2) from section to section, and (3) from section to sentence.

To update section nodes with their neighbor sentence nodes, GoSum employs the GAT and feed-forward net (FFN) layers as follows:

$$U_{s \to c} = \text{GAT}(H_c^0, H_s^0, H_s^0) \tag{15}$$

$$H_c^1 = \text{FFN}(U_{s \to c} + H_c^0) \tag{16}$$

In the above equations, H_s^0 is the initialized representation of sentence nodes, and H_c^0 represents the initialized representation of section nodes. $\text{GAT}(H_c^0, H_s^0, H_s^0)$ denotes H_c^0 as an attention query, and H_s^0 as a key and value. We then proceed to update section nodes based

on their edge connections:

$$U_{c\to c} = \text{GAT}(H_c^1, H_c^1, H_c^1) \tag{17}$$

$$H_c^2 = \text{FFN}(U_{c\to c} + H_c^1) \tag{18}$$

Once a section node is updated, it acquires section-level discourse information. This information is then passed to the corresponding sentence nodes:

$$U_{c\to s} = \text{GAT}(H_s^0, H_c^2, H_c^2) \tag{19}$$

$$H_s^1 = \text{FFN}(U_{c\to s} + H_s^0) \tag{20}$$

Since GoSum uses only one-layer GAT, the output after the GAT operation is denoted as H_s^1. We then assign the value of H_s^1 to $H_{discourse}$.

3.3.2 Global context encoder

After the message passing phase, a BiLSTM takes H_s^1 as input and generates sentence embeddings H_s^g, which encode global contextual information:

$$H_s^g = \text{BiLSTM}(H_s^1) \tag{21}$$

This module captures global contextual information such as the sentence's position within the document and information from neighboring sentences, to create a comprehensive representation H_{global} of the sentence.

3.3.3 Extraction history encoder

The concept of using extraction history features to sequentially extract sentences was first introduced in NeuSum [12]. The purpose of this approach is to avoid extracting sentences that contain redundant information. This method divides the extracted sentences from the remaining unextracted sentences and uses an extraction history encoder (EHE) to generate feature representations for the remaining sentences. These features are then used to guide the scoring of unextracted sentences.

Assume that at the current extraction step t, t sentences have been extracted to constitute the summary, and the model is preparing to extract the next sentence from the remaining $n-t$ sentences. The initial representation of sentences encoded with extraction history features borrows from the global contextual features obtained for each sentence, represented as H_s^g. The features for the two types of sentences are separately represented as the initial representation of extracted sentences H_{ext}^0, and the initial representation of remaining sentences to be extracted H_{rem}^0. The representation among unextracted sentences is first calculated using a self-attention [15] layer:

$$H_{rem}^1 = \text{SelfAttention}_1(H_{rem}^0, H_{rem}^0, H_{rem}^0) \tag{22}$$

Subsequently, another self-attention layer allows the unextracted sentences to attend to the already extracted sentences:

$$H_e^t = \text{SelfAttention}_2(H_{ext}^0, H_{ext}^0, H_{rem}^1) \tag{23}$$

Through this process, after t sentences have been extracted, the sentences that have not yet been extracted obtain the extraction history feature H_e^t. We then set $H_{history} = H_e^t$. This feature can capture relevant information from the extraction history, helping to reduce redundancy in the sentences extracted.

3.4 Sentence extractor

To determine whether to stop extraction, a multi-head pooling followed by a multilayer perceptron (MLP) is used to compute the stop signal based on the representation in Eq. 9. Furthermore, an additional MLP is employed to determine the optimal sentence for extraction:

$$p_{stop} = \text{MLP}_{stop}(H_{discourse}, H_{global}, H_{history}) \tag{24}$$

$$p_{score} = \text{MLP}_{score}(H_{discourse}, H_{global}, H_{history}) \tag{25}$$

If p_{stop} exceeds a certain threshold, then the model stops extracting sentences.

Algorithm 1 Training procedure of GoSum

Require: Document-Summary pair $< D, Y >$

An oracle label sequence $\hat{X} = \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_T\}$ is already sampled using beam search from D and Y, with corresponding summary $\hat{Y}$ having ROUGE scores r against Y.

Get discourse-aware sent embedding H_s^1:
  Initialize Graph H_s^0, H_c^0
  Message passing $(H_s^0, H_c^0) \rightarrow H_c^1$
  Message passing $(H_c^1, H_c^1) \rightarrow H_c^2$
  Message passing $(H_c^2, H_s^0) \rightarrow H_s^1$

Get global-content sent embedding H_s^g
for $t \leftarrow 1$ to T **do**
  Produce extraction history embedding H_e^t for the remaining sentences.
  Output the probability of the sentence from the extractor to select y_t and p_{stop} by using state $S_t = [H_s^1, H_s^g, H_e^t]$.
  get action likelihood $\pi(A_t|S_t, \theta)$ by Eq. 8
  Update policy gradient: $\theta \leftarrow \theta + l \cdot r \nabla \pi(A_t|S_t, \theta)$
  $t \leftarrow t + 1$
end for

3.5 Offline reinforcement learning

Reinforcement learning approximates the policy network distribution in Eq. (5) by sampling state-action mapping from the policy network. The training samples for online reinforcement learning algorithms are acquired by sampling from the currently trained policy net. However, this approach requires additional sampling time and often yields low-quality sampled examples, which can slow down model convergence.

In this paper, we employ offline reinforcement learning. During sampling, we retain the top-k labels with the highest ROUGE scores obtained through beam search. This strategy enables the policy network distribution in Eq. 5 to approximate the oracle distribution [39], facilitating quicker convergence and improved performance. After integrating with the graph neural network, the training process of GoSum for each training sample $< D, Y >$ is illustrated in Algorithm 1.

Table 1 The datasets used in the experiments

	PubMed	arXiv
avg. # sents of document	89	207
avg. # sents of summary	6.8	9.8
avg. # tokens of document	2730	5206
avg. # tokens of summary	181	238
avg. # sections of document	6.0	5.5
# Train	116 937	202 880
# Valid	6633	6436
# Test	6658	6440

4 Experiments

4.1 Summarization Datasets

We evaluate our model on the two long scientific paper summarization datasets: PubMed and arXiv [44]. Both datasets provide structural information for the input papers. The input for these two datasets consists of the full text of research papers (excluding the abstracts), and the target summary is the abstract of the papers. The statistics of the datasets are represented in Table 1.

4.2 Experimental setup

4.2.1 Evaluation metrics

The model performance is evaluated using the ROUGE score [10]. We report the F1 score for unigram, and bigram overlap (ROUGE-1, ROUGE-2), as well as the F1 score for the longest common subsequence (ROUGE-L).

4.2.2 Implementation details

GoSum and its variants are trained on four TITAN XP GPUs. Our model is trained using the Adam [45] optimizer with a learning rate of 0.0001, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. Since the original PubMed and arXiv datasets do not provide training labels for extractive summarization, we employ beam search to obtain extractive Oracle summary labels. For each document–abstract pair, we generate a maximum of 15 different oracle labels.

GoSum and its variants are trained for 20 epochs on both the PubMed and arXiv datasets. In each iteration, for each input document, we randomly sample one pre-sampled Oracle label for training. Model checkpoints are saved and evaluated every 10,000 steps. During the testing phase, the threshold for p_{stop} is set to 0.6 for the PubMed dataset and 0.45 for the arXiv dataset. The maximum sequence length for extracted summaries is set to 7 for the PubMed dataset and 6 for the arXiv dataset. The experiments of parameter selection are discussed in Sect. 4.5.4.

4.2.3 Baselines

We compare GoSum with several extractive baselines on the two datasets mentioned above. The baselines include:

- **Local–Global** [22]: A BiLSTM-based model that incorporates local and global contexts to extract summaries. **Local–Global+RdLoss** [36]: Extends local–global by incorporating a redundancy reinforcement learning loss.
- **HEROS** [24]: A graph-based approach with nodes from different discourse levels. It first applies a content ranking module to filter salience sentences. To ensure that the input is consistent with other baselines, we also record the results without the content ranking module.
- **Topic-GraphSum** [30]: Uses BERT as the document encoder and applies GAT to encode the topic-sentence graph.
- **Hi-struct+** [26]: Uses Longformer to first encode the entire document, and then applies transformers inject hierarchical structure information explicitly.
- **NeuSum** [12]: Considers the extraction history when jointly scoring and selecting sentences.
- **MemSum** [37]: A RL-based extractive summarizer that adopts BiLSTM to encode input sentences.
- **Sent-CLF** and **Sent-PTR** [46]: LSTM-based sentence classifier and a hierarchical seq2seq sentence pointer, respectively.
- **Lodoss** [25]: Use Longformer to get token embeddings, and employ a multi-task learning approach that improves sentence representation by simultaneously performing text summarization and section segmentation.

We evaluate the performance of these models on the PubMed and arXiv datasets.

4.3 Performance comparisons

Table 2 reports the results of our model on the arXiv and PubMed datasets. On both datasets, GoSum outperforms state-of-the-art extractive models in terms of ROUGE-1 and ROUGE-L scores. Specifically, on the PubMed dataset, the R-1 and R-L metrics surpass Lodoss by 0.45 and 0.36, respectively. However, GoSum lags behind Lodoss in the R-2 metric. According to the corresponding paper [25], this might be because Lodoss included a DPP regularizer during training, which encourages the selection of a set of salient and diverse sentences for the summary.

RL-based methods such as GoSum, MemSum, and LG-RdLoss demonstrate substantial performance gains, highlighting the effectiveness of reinforcement learning. Among these methods, GoSum outperforms MemSum. On the PubMed and arXiv datasets, the (R-1/R-2/R-L) metrics are improved by (0.16/0.23/0.26) and (0.58/0.62/0.68), respectively. This improvement can be attributed to two factors: (1) the use of the structural information from the input articles and (2) the use of graphs to model sentences and sections. By incorporating structural information, sentences can access more comprehensive information from sections, while sections can share and propagate topical information.

It is worth noting that GoSum demonstrates a greater performance improvement on the PubMed dataset compared to the arXiv dataset. One possible reason for this difference is the higher accuracy of the section information provided by the PubMed dataset, as discussed in detail in Sect. 4.5.3.

Table 2 Results on arXiv and PubMed dataset

Models	arXiv R-1	R-2	R-L	PubMed R-1	R-2	R-L
Oracle	60.00	30.60	53.03	61.99	34.95	56.76
Lead-10	30.52	10.33	31.44	37.45	14.19	34.07
Local–Global [22]	43.77	17.50	38.71	45.18	20.20	40.72
+ RdLoss [36]	44.01	17.79	39.09	45.30	20.42	40.95
Sent-CLF [46]	34.01	8.71	30.41	45.01	19.91	41.16
Sent-PTR [46]	42.32	15.63	38.06	43.30	17.92	39.47
HEROS [24]	47.74	20.46	42.39	48.14	21.82	43.33
w/o content ranking	45.90	18.33	40.78	46.63	20.63	42.01
Topic-GraphSum [30]	46.05	19.97	33.61	48.85	21.76	35.19
Hi-struct+ [26]	44.94	17.42	39.90	46.59	20.39	42.11
NeuSum [12]	–	–	–	47.46	21.92	42.87
MemSum [37]	48.42	20.30	42.54	49.25	22.94	44.42
Lodoss [25]	48.45	**20.72**	42.55	49.38	**23.89**	44.84
GoSum (ours)	**48.61**	20.53	**42.80**	**49.83**	23.56	**45.10**

Bold represent the best results, while the numbers with underlines represent the second-best results

Table 3 Ablation studies of GoSum on the PubMed dataset

	ROUGE-1	ROUGE-2	ROUGE-L	Avg.Time(ms)
GoSum	49.83	23.56	45.10	580.9
GoSum with different graph design				
w/o sec2sec edges	49.72	23.46	44.99	562.5
w/o graph	49.44	23.24	44.74	333.2
w/o sec & graph	49.22	23.02	44.49	316.2
GoSum with different state representation				
w SecE	49.80	23.56	45.08	595.7
w/o GCE	48.80	22.34	44.16	545.8
w/o DLE	48.01	21.84	43.57	303.0
w/o EHE	49.24	23.09	44.28	527.1
GoSum with different RL setting				
w/o reward	49.64	23.37	44.96	–
Sample top-1	49.10	23.00	44.42	–
Sample top-2	49.27	23.07	44.61	–
Sample top-4	49.64	23.33	44.96	–

4.4 Ablation studies

In Table 3, we conduct ablation studies by comparing GoSum with its variants. To validate the performance of the graph structure, we implement the following GoSum variants:

- **GoSum w/o sec2sec edges**: This variant removes section-to-section edges in graph construction and takes H_c^1 as a key input in Eq. 17.

- **GoSum w/o graph**: This variant does not include graph modeling. Specifically, the global contextual embedding H_s^g is obtained directly using H_s^0. The state representation S_t in Eq. 9 includes one additional embedding H_c^0 to capture section information.
- **GoSum w/o sec & graph**: This variant does not use document structural information and graph modeling.

The improvements observed from **GoSum w/o graph** to **GoSum w/o sec2sec edges** demonstrate that the inclusion of paper structure information can slightly enhance the performance of GoSum. However, the performance of GoSum shows a more significant improvement when using graphs to model the relationships between sentences and sections.

Next, we examine the effects of different embeddings on the performance of GoSum. For **GoSum w SecE**, the extractor incorporates an additional section representation H_c^2 in Eq. (9). **GoSum w/o GCE**, **GoSum w/o DLE**, and **GoSum w/o EHE** remove the Global context embedding H_{global}, the discourse-aware sentence embedding $H_{discourse}$, and the extraction history embedding $H_{history}$ in Eq. (9), respectively. The results are presented in Table 3. Although **GoSum w SecE** includes an extra embedding, the resulting scores instead exhibit a slight decrease. This indicates that the information about section nodes has already been incorporated into the local content embedding during the graph update process, making the addition of section embedding redundant and potentially leading to over-fitting. If the other three embeddings are removed, the performance of GoSum drops. Among them, **GoSum w/o DLE**, which removes $H_{discourse}$, shows the most significant decrease in performance. This finding also suggests that the discourse-aware local sentence embedding contains valuable information that contributes to the overall performance of GoSum.

In Table 2, we also recorded the running times of GoSum and its different variants. In the single-card Titan XP environment for inference, GoSum requires an average of 580 ms to generate a summary for a paper. At the same time, we found that graph computation accounts for a significant portion of GoSum's inference time. Models that do not use graph structures, such as **GoSum w/o DLE** or **GoSum w/o graph**, have inference times of around 300 ms.

4.5 Results analysis

4.5.1 Impact of reinforcement learning: sampling and reward mechanisms

There are two factors in reinforcement learning that contribute to the improved performance of GoSum. First, RL enables more high-quality oracle label sampling equivalent to data augmentation, providing the model with additional training instances. Second, RL furnishes feedback rewards to different samples during training, aiding in differentiating between favorable and unfavorable instances. The experimental analysis investigating the influence of these two factors on the performance of GoSum is reported in Table 3. The comparison methods are set as follows:

- **Complete RL**: Original training setting of GoSum. An average of 6.52 oracle labels are sampled per document–abstract pair.
- **w/o reward**: With rewards set to 1 for all samples, there is a slight decrease in performance compared to the complete RL model.
- **sample top-k**: Model is trained using only the k highest sampled label sequences of an input document.

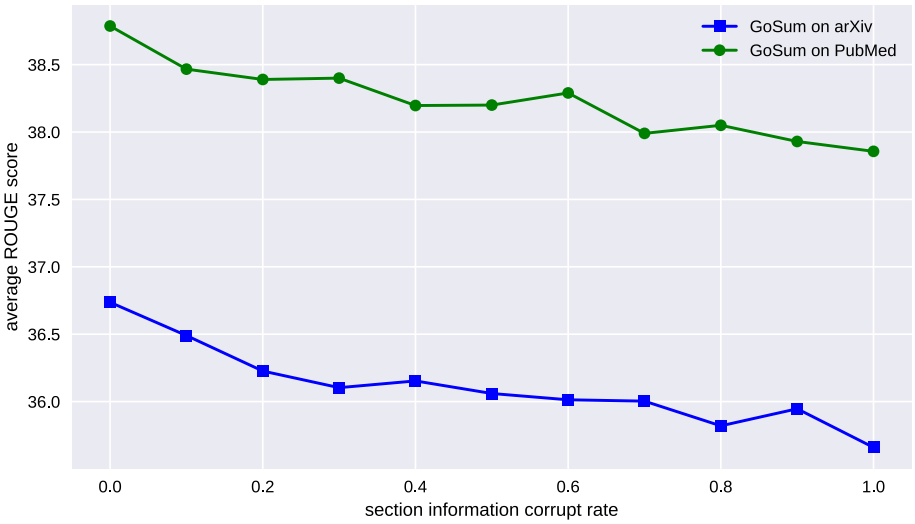


Fig. 2 The performance of GoSum is evaluated by corrupting the section information of the input sentences at different rates, as shown on the x-axis. The average ROUGE score is plotted on the y-axis. The green dots represent the scores of GoSum on the PubMed dataset, while the blue dots represent the results of GoSum on the arXiv dataset

Observably, as the number of samples increases, the performance of GoSum exhibits significant improvement. In summary, the experimental findings pertaining to reinforcement learning validate our initial conjecture.

4.5.2 Impact of graph-organized discourse states: section information

The integration of reinforcement learning and graph neural networks in GoSum has led to significant performance improvements compared to other extractive approaches. However, one might naturally question why GNN can enhance GoSum. There could be two reasons:

First, the use of a graph enables the capture of the input discourse information. The graph structure facilitates the representation and understanding of relationships between sentences and sections in the document.

Second, the hierarchical nature of the graph is crucial for information diffusion, with section nodes serving as reservoirs for the information from their respective sentences. To validate these hypotheses, we conduct experiments to evaluate the performance of GoSum, and we gradually disrupt the section attribution information of the input sentences while keeping the other model parameters unchanged. This disruption of section attribution information may obscure the discourse information, but the hierarchical structure of the graph remains intact.

Specifically, we gradually disrupt the section attribution of the input sentences, with a 10% increment in each experiment. Due to time constraints, we select 10,000 samples from each of the PubMed and arXiv datasets for training. As shown in Fig. 2, the performance of GoSum exhibits a rapid decline as the amount of discourse information decreases. However, due to the limited number of training samples and the inherent instability of reinforcement learning, the performance of GoSum fluctuates slightly between datasets but overall shows a gradually decreasing trend. The initial significant drop in GoSum's performance indicates

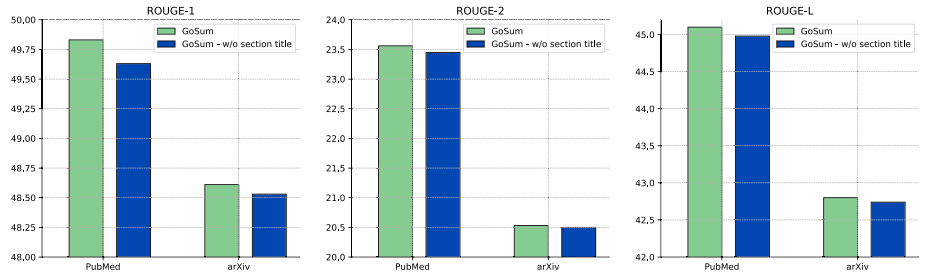


Fig. 3 Comparisons between GoSum and those trained without the section titles

its sensitivity to the accuracy of section information. This further confirms that precise and reliable discourse information is crucial for enhancing the performance of GoSum.

4.5.3 Impact of section titles: semantic cues and quality

In addition to the discourse information present in the literature, which organizes sentences into various sections, section-specific titles such as "introduction," and "methodology" also carry semantic information that contributes to improving the performance of GoSum. To investigate the impact of section titles, we establish a control model called **GoSum w/o SecTitle**. This model mirrors the architecture of GoSum, but replaces section titles in the data with meaningless placeholders like "section #id" during the training. The experimental results presented in Fig. 3 reveal that the performance of **GoSum w/o SecTitle** is slightly worse than that of GoSum. This suggests that while not indispensable, the semantic cues provided by section titles are indeed beneficial. However, the primary factor contributing to performance improvement lies in the discourse hierarchies within the documents. Moreover, the decline in performance of **GoSum w/o SecTitle** is more significant on the PubMed dataset. The discrepancy in the performance between **GoSum w/o SecTitle** and GoSum on the arXiv dataset is not significant, possibly because the document title quality in the arXiv dataset does not consistently meet the required quality standards.

4.5.4 Impact of extracted summary length

The length of the output summary influences the ROUGE scores in extractive summarization, primarily by controlling the number of extracted sentences. GoSum utilizes two related parameters to control the number of extracted sentences. The threshold p_{stop} is an important parameter that determines whether GoSum stops extracting sentences. To determine the optimal threshold, we conducted experiments on the validation sets of PubMed and arXiv, sampling thresholds from 0.1 to 0.9 at intervals of 0.1, and between 0.4 and 0.7 at intervals of 0.05. The experimental results are depicted in Fig. 5. Another parameter controlling the length of extracted sentences is the maximum limit of extracted sentences k_{max}. Figure 4 illustrates the experimental results of GoSum under the ROUGE metrics with $k_{max} \in \{5, 6, 7, 8\}$.

4.6 Semantic evaluation

ROUGE scores solely measure the n-gram overlap between sentences. Therefore, in this section, we delve deeper by conducting both manual and automated semantic evaluations

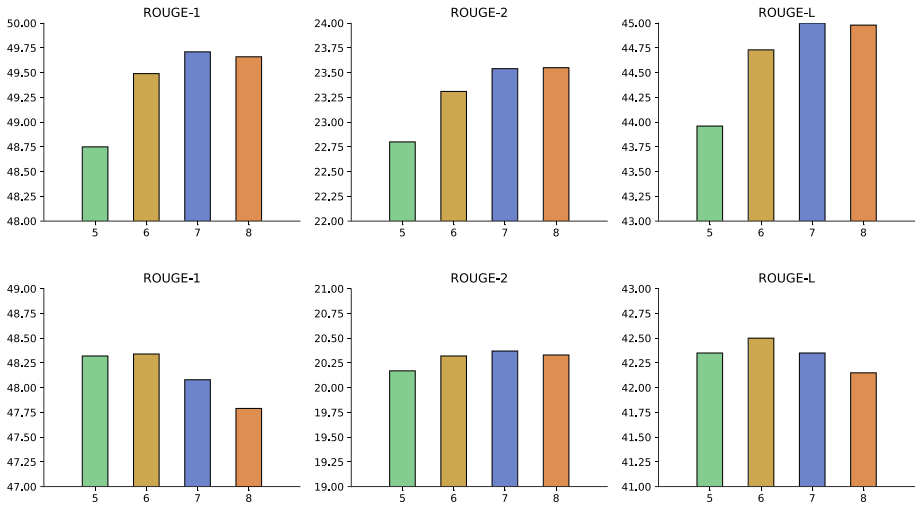


Fig. 4 On the PubMed(above) and arXiv(bottom) dataset, the results of different maximum limitations k_{max} for sentence extraction

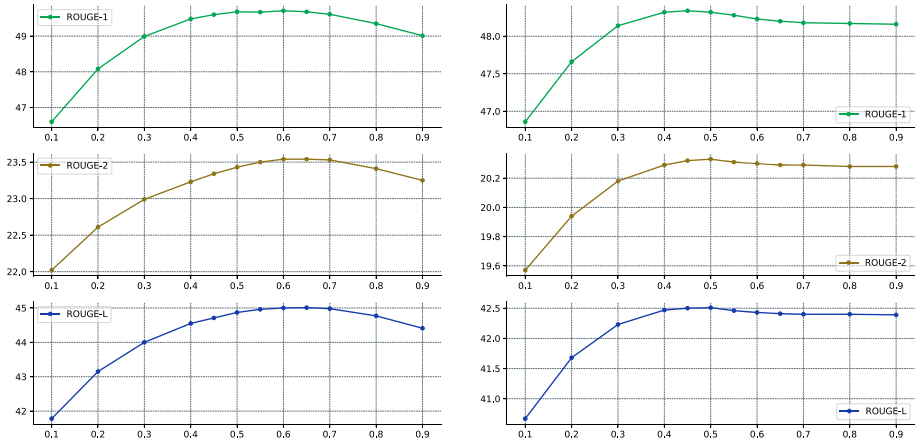


Fig. 5 The ROUGE scores for different threshold of p_{stop} on PubMed(left) and arXiv(right) validation split

to provide a more comprehensive analysis and comparison of GoSum's performance. In addition, we choose MemSum as the baseline. MemSum also employs reinforcement learning and extraction history encoding and has shown excellent performance in long document summarization.

Human evaluations involve two annotators who assess the performance of GoSum and MemSum based on two criteria: **less redundancy** and **coverage**. They evaluate 50 randomly selected PubMed documents. The annotators determine which model performs better or if they are equivalent. The results of the manual evaluation are as follows:

- In terms of less redundancy: GoSum and MemSum rank 1.40 and 1.60, respectively.
- In terms of coverage: GoSum and MemSum rank 1.52 and 1.48, respectively.

Paper Title: A Case of Septic Arthritis of Shoulder Presenting as Stiffness of the Shoulder

Golden Summary	Introduction: septic arthritis of the shoulder is uncommon in adults. It is a surgical emergency as joint destruction occurs rapidly and can lead to significant morbidity and mortality. Accurate diagnosis can be particularly challenging in patients with underlying liver disease. Mri is a useful adjunct in early detection of atypical causes of shoulder pain. Case report: a 43 years old male came to our outpatient department with complaints of pain and stiffness of his left shoulder. On examination, his shoulder movements were severely restricted. Further evaluation with mri revealed septic arthritis of left gleno - humeral joint for which emergency arthroscopic debridement was done. Conclusion: septic arthritis of shoulder may not present with classical clinical features. Hence, a through clinical and radiological evaluation will help us prognosticate and treat accordingly thereby preventing complications like septic shock, osteomyelitis.
Extraction Overlap Between Memsum & GoSum	Septic arthritis of the shoulder is uncommon in adults. It is a surgical emergency as joint destruction occurs rapidly and can lead to significant morbidity and mortality. A 43 years old male came to our outpatient department with complaints of pain and stiffness of his left shoulder. On examination, his shoulder movements were severely restricted. Further evaluation with mri revealed septic arthritis of left gleno - humeral joint for which emergency arthroscopic debridement was done. Septic arthritis of shoulder may not present with classical clinical features. Hence, a through clinical and radiological evaluation will help us prognosticate and treat accordingly thereby preventing complications like septic shock, osteomyelitis.
MemSum	We report an unusual presentation of shoulder septic arthritis in a 43 years old man with no other clinical signs and symptoms of classical septic arthritis. ROUGE score with golden summary R-1: 84.32 R-2: 78.19 R-L: 81.34
GoSum	None ROUGE score with golden summary： R-1: 87.60 R-2: 85.83 R-L: 87.60

Fig. 6 An example of summaries by MemSum and GoSum

The evaluation results indicate that GoSum demonstrates lower redundancy and comparable coverage compared to MemSum. It is observed that GoSum tends to extract sentences that contain specialized terminologies and rich information, rather than opting for more conservative and generalized sentences.

Moreover, we also evaluated GoSum and MemSum on the PubMed dataset's test set using BERTScore [47]. The corresponding BERTScore values for the two methods are 86.57% and 86.43%, respectively. This indicates that GoSum is closer to the golden summary in terms of semantic similarity.

4.7 Case study and error analysis

To facilitate a more intuitive comparison between GoSum and other baselines, we present an example from the PubMed dataset in Fig. 6. The row called **Overlap** indicates the sentences extracted by both methods, while the sentences extracted by each method individually are presented in their corresponding table rows.

In Table 6, the vast majority of sentences predicted by GoSum and MemSum are identical, summarizing the main content of the gold abstract. However, MemSum additionally predicts a sentence related to patient reports (highlighted in the table), which exhibits some redundancy with the other predicted sentences. In contrast, GoSum does not predict any extra sentences and performs better in terms of redundancy.

We also conduct error analysis in Figs. 7 and 8. We found that the errors in GoSum's output results can be categorized into the following five types:

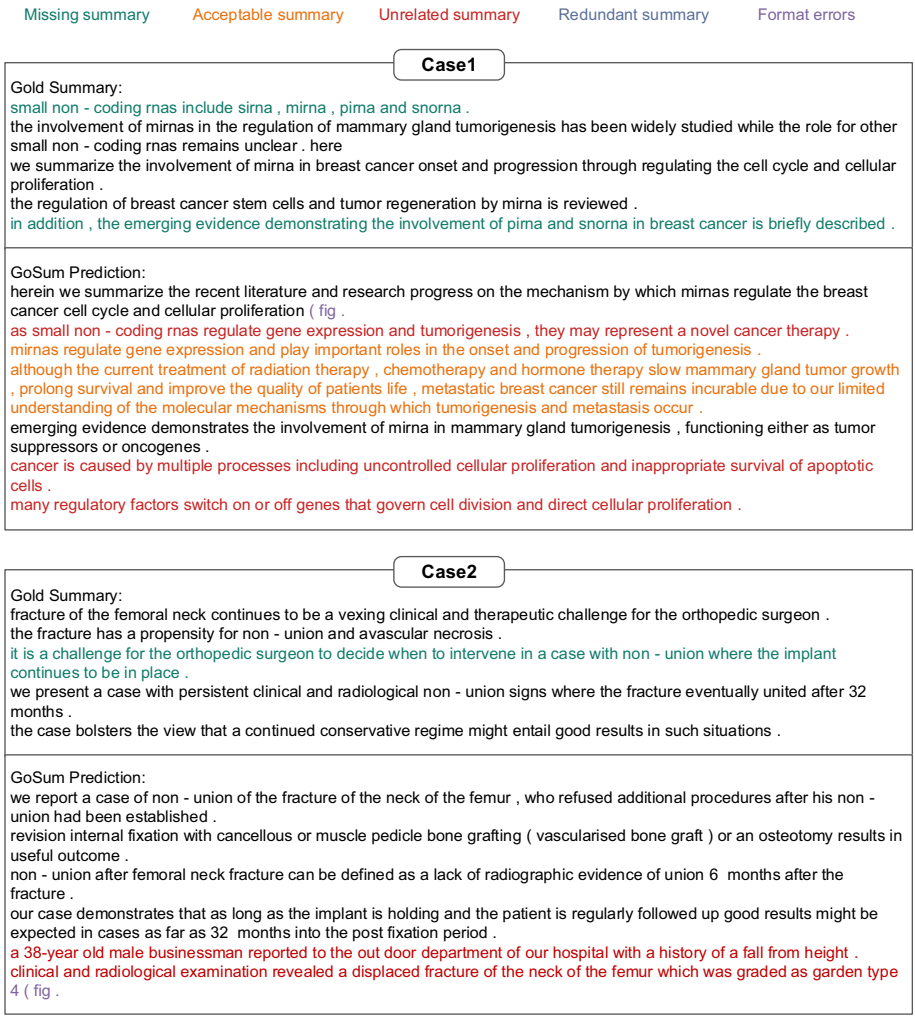


Fig. 7 Case 1 and Case 2 of error analysis

1. Formatting and grammatical errors: Since the extracted sentences come from the original text of the paper, there may be words or abbreviations like “fig.”, “table.”, “etc.”, that do not conform to the summary format.
2. Missing key points in the Gold summary: Some summarization points in the Gold summary are not captured by the model.
3. Unrelated summary: The summary output by the model is unrelated to both the gold summary and the overall content of the paper.
4. Acceptable summary: The summary output by the model is not strongly related to the gold summary, but it can also be considered a reasonable summary when starting from the original text of the paper.
5. Redundant summary: A sentence in the model’s output is similar to other sentences in the output, resulting in redundancy.

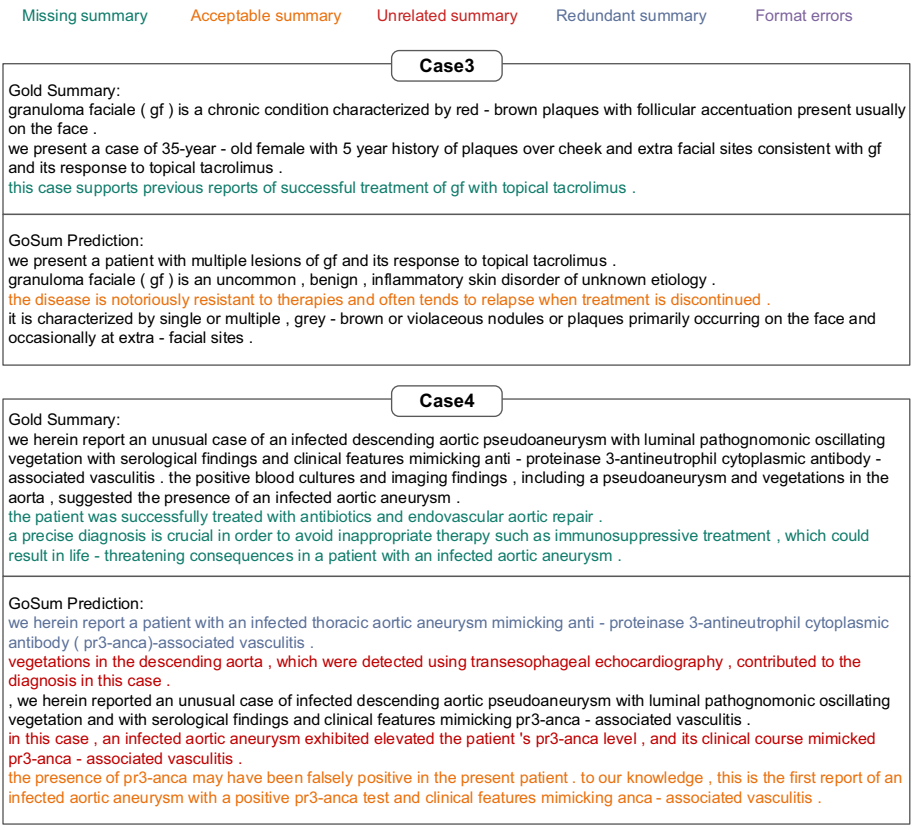


Fig. 8 Case 3 and Case 4 of error analysis

5 Conclusion

In this paper, we present GoSum, a novel approach for summarizing long documents by effectively integrating reinforcement learning with a graph neural network. Our experiments on the arXiv and PubMed datasets demonstrate that GoSum achieves state-of-the-art performance. Through the ablation studies, we further analyze the impact of discourse information on GoSum. The results reveal that the effectiveness of GoSum is attributed to the hierarchical organization of sentences and the semantic cues provided by section titles in the documents.

However, GoSum relies on input documents having section structure information. Thus, it may struggle when applied to documents lacking this structure. To address this limitation, our future work aims to explore methods for automatically generating discourse information from documents.

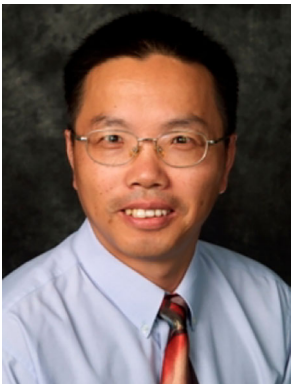
Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant No. 62272105), Shanghai Municipal Science and Technology Major Project (Grant No. 2018SHZDZX01), ZJ Lab and Shanghai Center for Brain Science and Brain-Inspired Intelligence Technology and the 111 Project.

References

1. Xiong W, Gupta A, Toshniwal S, Mehdad Y, Yih W-t (2022) Adapting pretrained text-to-text models for long text sequences. arXiv:2209.10052
2. Pang B, Nijkamp E, Kryściński W, Savarese S, Zhou Y, Xiong C (2023) Long document summarization with top–down and bottom-up inference. In: Findings of the association for computational linguistics: EACL 2023, pp 1237–1254
3. Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A et al (2020) Language models are few-shot learners. Adv Neural Inf Process Syst 33:1877–1901
4. OpenAI (2023) Gpt-4 technical report. arXiv:2303.08774
5. Touvron H, Lavril T, Izacard G, Martinet X, Lachaux M-A, Lacroix T, Rozière B, Goyal N, Hambro E, Azhar F, et al (2023) Llama: open and efficient foundation language models. arXiv:2302.13971
6. Chung HW, Hou L, Longpre S, Zoph B, Tay Y, Fedus W, Li Y, Wang X, Dehghani M, Brahma S et al (2022) Scaling instruction-finetuned language models. arXiv:2210.11416
7. Workshop B, Scao TL, Fan A, Akiki C, Pavlick E, Ilić S. Hesslow D, Castagné R, Luccioni AS, Yvon F, et al (2022) Bloom: a 176b-parameter open-access multilingual language model. arXiv:2211.05100
8. Kryściński W, McCann B, Xiong C, Socher R (2020) Evaluating the factual consistency of abstractive text summarization. In: Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP), pp 9332–9346
9. Pagnoni A, Balachandran V, Tsvetkov Y (2021) Understanding factuality in abstractive summarization with frank: a benchmark for factuality metrics. In: Proceedings of the 2021 Conference of the North American chapter of the association for computational linguistics: human language technologies, pp 4812–4829
10. Lin C-Y (2004) ROUGE: a package for automatic evaluation of summaries. In: Text summarization branches out, pp 74–81. https://www.aclweb.org/anthology/W04-1013 Accessed 27 July 2019
11. Zhong M, Liu P, Chen Y, Wang D, Qiu X, Huang X (2020) Extractive summarization as text matching. In: Proceedings of the 58th annual meeting of the association for computational linguistics, pp 6197–6208. https://doi.org/10.18653/v1/2020.acl-main.552. Accessed 21 Dec 2020
12. Zhou Q, Yang N, Wei F, Huang S, Zhou M, Zhao T (2018) Neural document summarization by jointly learning to score and select sentences. In: Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: long papers), pp 654–663
13. Hermann KM, Kocisky T, Grefenstette E, Espeholt L, Kay W, Suleyman M, Blunsom P (2015) Teaching machines to read and comprehend. Adv Neural Inf Process Syst 28
14. Koupaee M, Wang WY (2018) Wikihow: a large scale text summarization dataset. arXiv:1810.09305
15. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Adv Neural Inf Process Syst 30
16. Narayan S, Cohen SB, Lapata M (2018) Ranking sentences for extractive summarization with reinforcement learning. In: Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long papers), pp 1747–1759
17. Liu Y, Lapata M (2019) Text summarization with pretrained encoders. In: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), pp 3730–3740
18. Zhang J, Zhao Y, Saleh M, Liu PJ (2020) PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. arXiv:1912.08777
19. Zaheer M, Guruganesh G, Dubey KA, Ainslie J, Alberti C, Ontanon S, Pham P, Ravula A, Wang Q, Yang L et al (2020) Big bird: transformers for longer sequences. Adv Neural Inf Process Syst 33:17283–17297
20. Beltagy I, Peters ME, Cohan A (2020) Longformer: the long-document transformer. arXiv:2004.05150
21. Huang L, Cao S, Parulian N, Ji H, Wang L (2021) Efficient attentions for long document summarization. In: Proceedings of the 2021 conference of the north American chapter of the association for computational linguistics: human language technologies, pp 1419–1436
22. Xiao W, Carenini G (2019) Extractive summarization of long documents by combining global and local context. In: EMNLP-IJCNLP, pp 3011–3021
23. Collins E, Augenstein I, Riedel S (2017) A supervised approach to extractive summarisation of scientific papers. In: Proceedings of the 21st conference on computational natural language learning (CoNLL 2017), pp 195–205
24. Zhu T, Hua W, Qu J, Zhou X (2021) Summarizing long-form document with rich discourse information. In: Proceedings of the 30th ACM international conference on information & knowledge management, pp 2770–2779

25. Cho S, Song K, Wang X, Liu F, Yu D (2022) Toward unifying text segmentation and long document summarization. In: Proceedings of the 2022 conference on empirical methods in natural language processing, pp 106–118
26. Ruan Q, Ostendorff M, Rehm G (2022) Histruct+: improving extractive text summarization with hierarchical structure information. In: Findings of the association for computational linguistics: ACL 2022, pp 1292–1308
27. Erkan G, Radev DR (2004) Lexrank: graph-based lexical centrality as salience in text summarization. J Artific Intell Res 22:457–479
28. Wang D, Liu P, Zheng Y, Qiu X, Huang X (2020) Heterogeneous graph neural networks for extractive document summarization. In: Proceedings of the 58th annual meeting of the association for computational linguistics, pp 6209–6219
29. Jia R, Cao Y, Tang H, Fang F, Cao C, Wang S (2020) Neural extractive summarization with hierarchical attentive heterogeneous graph network. In: Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP), pp 3622–3631
30. Cui P, Hu L, Liu Y (2020) Enhancing extractive text summarization with topic-aware graph neural networks. In: Proceedings of the 28th international conference on computational linguistics, pp 5360–5371
31. Xu J, Gan Z, Cheng Y, Liu J (2020) Discourse-aware neural extractive text summarization. In: Proceedings of the 58th annual meeting of the association for computational linguistics, pp 5021–5031
32. Wu Y, Hu B (2018) Learning to extract coherent summary via deep reinforcement learning. In: Proceedings of the AAAI conference on artificial intelligence, vol 32
33. Böhm F, Gao Y, Meyer CM, Shapira O, Dagan I, Gurevych I (2019) Better rewards yield better summaries: Learning to summarise without references. In: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), pp 3110–3120
34. Gao Y, Meyer CM, Gurevych I (2018) April: interactively learning to summarise by combining active preference learning and reinforcement learning. In: Proceedings of the 2018 conference on empirical methods in natural language processing, pp 4120–4130
35. Chen Y-C, Bansal M (2018) Fast abstractive summarization with reinforce-selected sentence rewriting. In: Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: long papers), pp 675–686
36. Xiao W, Carenini G (2020) Systematically exploring redundancy reduction in summarizing long documents. In: Proceedings of the 1st conference of the Asia–Pacific chapter of the association for computational linguistics and the 10th international joint conference on natural language processing, pp 516–528
37. Gu N, Ash E, Hahnloser R (2022) Memsum: extractive summarization of long documents using multi-step episodic markov decision processes. In: Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers), pp 6507–6522
38. Pang RY, He H (2020) Text generation by learning from demonstrations. In: International conference on learning representations
39. Xu Y, Lapata M (2022) Text summarization with oracle expectation. In: The eleventh international conference on learning representations
40. Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learn 8(3):229–256
41. Pennington J, Socher R, Manning CD (2014) Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1532–1543
42. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
43. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2017) Graph attention networks. arXiv:1710.10903
44. Cohan A, Dernoncourt F, Kim DS, Bui T, Kim S, Chang W, Goharian N (2018) A discourse-aware attention model for abstractive summarization of long documents. In: Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: human language technologies, volume 2 (short papers), pp 615–621. https://doi.org/10.18653/v1/N18-2097 . Accessed 2020-10-12
45. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: ICLR (Poster)
46. Pilault J, Li R, Subramanian S, Pal C (2020) On extractive and abstractive neural document summarization with transformer language models. In: Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP), pp 9308–9319
47. Zhang T, Kishore V, Wu F, Weinberger KQ, Artzi Y (2019) Bertscore: evaluating text generation with bert. In: International conference on learning representations

Junyi Bian is currently a fourth-year Ph.D. student at Fudan University, having previously obtained a Master of Engineering degree in Computer Science from ShanghaiTech University. His research focuses primarily on information extraction within the field of natural language processing, especially in biomedical named entity recognition, relation extraction, and extractive summarization of literature.



Xiaodi Huang is an Associate Professor in the School of Computing, Mathematics, and Engineering at Charles Sturt University, Australia. He earned his Ph.D. degree in 2004. His research focuses on applied machine learning, visualization, and data analysis. Dr. Huang has published over 150 scholarly papers in international journals and conferences. Beyond his research contributions, Dr. Huang serves as an editor for various international journals and regularly assumes roles as a chair and committee member for numerous international conferences. He is a member of the ACM and a Senior Member of IEEE. For more information about his work and achievements, please visit his homepage at https://sites.google.com/view/xiaodih.

Hong Zhou is the Director of Intelligent Services at Wiley Partner Solution, leading AI strategies and the product development of intelligent services for research and publishing. He founded the AI R&D Team, Hong holds a Ph.D. in 3D modeling with AI and an MBA in Digital Transformation from Oxford. He is recognized as an AI thought leader in the scholarly publishing industry. He is Chef of the Scholarly Kitchen and co-chair of the AI group in ALPSP.

Tianyang Huang received the B.Sc. in statistics in Tongji University in 2022. He is currently pursuing the Ph.D. degree in Institute of Science and Technology for Brain-inspired Intelligence, Fudan University. His research interests mainly include the algorithm and application of extreme multi-label learning.

Shanfeng Zhu is a professor in the Institute of Science and Technology for Brain-inspired Intelligence, Fudan University, Shanghai, China. His research interests include machine learning, data mining, and their applications in text mining and bioinformatics.

Authors and Affiliations

Junyi Bian[1,8] · Xiaodi Huang[2] · Hong Zhou[3] · Tianyang Huang[4] · Shanfeng Zhu[4,5,6,7,8]

✉ Shanfeng Zhu
zhusf@fudan.edu.cn

Junyi Bian
20110240003@fudan.edu.cn

Xiaodi Huang
xhuang@csu.edu.au

Hong Zhou
hzhou@atypon.com

Tianyang Huang
tyhuang22@m.fudan.edu.cn

1 School of Computer Science, Fudan University, Shanghai 200433, China

2 School of Computing, Mathematics and Engineering, Charles Sturt University, Elizabeth Mitchell Dr, Albury, NSW 2640, Australia

[3] Atypon Systems, LLC, Oxford, UK

[4] Institute of Science and Technology for Brain-Inspired Intelligence, Fudan University, Shanghai 200433, China

[5] Key Laboratory of Computational Neuroscience and Brain-Inspired Intelligence, Fudan University, Shanghai 200433, China

[6] MOE Frontiers Center for Brain Science, Fudan University, Shanghai 200433, China

[7] Zhangjiang Fudan International Innovation Center, Fudan University, Shanghai 200433, China

[8] Shanghai Key Lab of Intelligent Information Processing, Fudan University, Shanghai 200433, China